FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Localização e Navegação de Robôs em Ambientes Dinâmicos e com Troca Automática de Baterias

Pedro Miguel Leão Guedes

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Supervisor externo: Engenheiro Jorge Paiva

Orientador: Professor Doutor António Paulo Moreira

Coorientador: Doutor Héber Sobreira

17 de Julho de 2018

Resumo

Para que um robô autónomo realize as tarefas a ele associadas com distinção, é fulcral que este possua um mecanismo de localização robusto e preciso, especialmente quando este navega em ambientes dinâmicos. Por outro lado, estes robôs estão associados a períodos de inatividade devido à necessidade de recarregar as suas baterias, diminuindo desta forma a sua rentabilidade. Em certas aplicações, como é o caso da vigilância, estes períodos "mortos" não são tolerados.

Assim, um dos objetivos da presente dissertação passa por dotar um robô móvel com um sistema de localização preciso e robusto. Atendendo ao objetivo, foram analisados os algoritmos de localização *Perfect Match* e AMCL, com o intuito de identificar em que situações é que estes algoritmos falhavam. De acordo com a análise realizada é proposto um mecanismo de supervisão e *tracking* da pose do robô, baseado essencialmente num mecanismo de deteção de falhas que recorre à técnica dos mínimos quadrados recursivos com fator de esquecimento exponencial. Tendo em conta os resultados obtidos, o mecanismo proposto conseguiu colmatar as falhas identificadas aquando da análise dos algoritmos de localizção e supervisão, provando ser uma mais valia para o aumento da robustez dos sistemas de localização em robôs móveis.

Já com o intuito de eliminar os períodos de inatividade associados aos robôs móveis, é proposto um mecanismo de troca automática de baterias, ou seja, um mecanismo em que não é necessária qualquer a intervenção humana. Para tal, foi projetada uma estação totalmente passiva, bem como todo o *software* necessário para o sistema de troca de baterias tendo em conta todos os requisitos do projeto. Mais precisamente, foi desenvolvido um algoritmo de deteção da estação de troca de baterias, assim como um conjunto de controladores que permitissem a realização de uma docagem precisa. Embora a estação projetada seja bastante simples, de acordo com os resultados obtidos, o sistema desenvolvido demonstrou possuir uma elevada repetibilidade e precisão, sendo esta inferior a 7 milímetros. Adicionalmente, o mecanismo de deteção da estação revelou ser bastante robusto face ao posicionamento do robô.

Abstract

In order for an autonomous robot to perform it's tasks with success, it's critical that the robot has a robust and precise localization mechanism for navigating in dynamic environments. On the other hand, these robots are associated with periods of inactivity due to the need to recharge their batteries, thus reducing their profitability. In certain applications, such as surveillance, these "dead"periods are not tolerated.

Thus, one of the objectives of this dissertation is to equip a mobile robot with a precise and robust localization system. Considering the objective, the Perfect Match and AMCL localization algorithms were analyzed, in order to identify in which situations these algorithms failed. According to the analysis, a mechanism for robot pose supervision and tracking is proposed, based essentially on a fault detection mechanism that uses the recursive least squares technique with an exponential forgetting factor. Considering the results obtained, the proposed mechanism was able to overcome the identified flaws at the momment of the algotithms analisys (localization and supervision algotithms), proving to be an added value for increasing the robustness of the localization systems in mobile robots.

Already in order to eliminate the periods of inactivity associated with mobile robots, a mechanism for automatic battery replacement is proposed, ie a mechanism in which no human intervention is necessary. For this, a fully passive docking station was designed, as well as all the software required for the battery exchange system taking into account all design requirements. More precisely, a battery exchange station detection algorithm has been developed, as well as a set of controllers that allow an accurate docking. Although the designed station is quite simple, according to the obtained results, the developed system has demonstrated a high repeatability and precision, being the precision less than 7 millimeters. In addition, the station detection mechanism proved to be quite robust in terms of robot positioning.

Agradecimentos

Em primeiro lugar, gostaria de agradecer aos meus orientadores A. Paulo Moreira e Héber Sobreira pelo apoio, pelas ideias e pela disponibilidade demonstrada para me ajudar ao longo de toda a minha dissertação. Gostaria também de deixar uma palavra de apreço à CLEVERHOUSE, em especial ao Eng. Jorge Paiva, por financiar e tornar possível todo o projeto associado a esta dissertação e ainda ao professor Paulo Costa pelas sugestões e ideias partilhadas.

Gostaria também de agradecer à equipa do INESC-TEC, em especial à Cláudia Rocha, ao Ivo Sousa e ao Eurico Sousa, pela disponibilidade e ajuda prestada sempre que precisei.

Aos meus amigos, André Pinto, Emanuel Pereira, Gonçalo Silva, Pedro Moura, Sandro Magalhães, Sérgio Pinto e Tiago Mendonça, um sincero obrigado pelos momentos que partilhamos, pelas piadas, pelas discussões saudáveis em busca de soluções para os nossos problemas e pelos momentos de descontração.

Gostaria de agradecer ao meu pai e à minha mãe por fazerem de mim o que sou hoje, e pelo enorme esforço que fizeram para que eu pudesse ter um curso superior. Obrigado por todo o carinho e apoio que demonstraram por mim. Ainda agradecer ao meu irmão também por todo o apoio prestado, não só ao longo desta dissertação, mas sim ao longo da minha vida.

Finalmente, à Diana Barros que sem dúvida foi um pilar muito importante na minha caminhada académica. Muito obrigado pelas tuas revisões, por me apoiares nos momentos difíceis, por não me deixares desanimar, e acima de tudo por estares sempre pronta para me ajudar em qualquer situação. A ti, aos meus pais e irmão, um especial e sincero OBRIGADO, sem vocês não teria conseguido certamente.

Pedro Guedes

"Work hard in silence.
Let your success be your noise."

Frank Ocean

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

F	Força
m	Massa
a	Aceleração
t	Tempo
MCL	Monte Carlo Localization
AMCL	Adaptative Monte Carlo Localization
SAMCL	Self-adaptative Monte Carlo Localization
SER	Similar Energy Region
IMCLRO	Improved Monte Carlo Localization with Robust Orientation estimation
PM	Perfect Match
ICP	Iterative Closest Point
RPROP	Resilient Propagation
EKF	Extended Kalman Filter
MbICP	Metric based Iterative Closest Point
PLICP	Point-to-line Iterative Closest Point
DFT	Discrete Fourier Transform
LRF	Laser Range Finder
LED	Light Emitting Diode
SLAM	Simultaneous Localization and Mapping

Capítulo 1

Introdução

1.1 Contexto

Com a evolução tecnológica, os robôs autónomos são requeridos e implementados nas mais diversas áreas, tendo várias aplicações tais como: monitorização de pacientes em hospitais [1, 2], guias de museu [3, 4], limpeza de interiores [5], transporte de materiais de um local para outro [6], entre outras.

Deste modo, estes robôs são alvo de investigação constante com o intuito de procurar soluções para os novos desafios. Um dos problemas atuais destes robôs diz respeito à sua localização e navegação, principalmente em ambientes dinâmicos, ou seja, onde as características do ambiente estão em constante alteração. Face a esta problemática, a comunidade científica tem uma missão acrescida relativamente à robustez da auto localização, uma vez que a complexidade dos algoritmos já utilizados é aumentada de forma a que o excesso de ruído de localização que os sensores detetam nestes meios, não degrade a precisão dos mesmos [7]. Assim sendo existe uma necessidade de evolução de algoritmos ao nível da localização e navegação em ambientes que englobam não só outros robôs, mas também seres humanos.

Outro problema em relação à navegação de robôs passa pela utilização da sua bateria. Por outras palavras, os robôs que já navegam de forma autónoma estão inerentemente associados a longos períodos de recarga, isto é, estão sujeitos a períodos de inatividade necessitando ainda da intervenção humana para o processo de recarga. Assim sendo, torna-se necessário fornecer soluções para que os robôs evitem ter períodos mortos e assim aumentar a sua rentabilidade, funcionando 24 horas por dia.

Após uma análise detalhada das necessidades que estão associadas aos robôs autónomos, surge a presente dissertação, que se intitula "Localização e Navegação de Robôs em Ambientes Dinâmicos e com Troca de Automática de Baterias". Neste sentido, esta dissertação será uma evolução da dissertação "Plataforma Robótica Genérica para robô de Logística, Serviços ou Vigilância com Mecanismo de Troca Automática de Baterias" elaborada pelo, agora mestre, Ivo Sousa.

1.2 Motivação

Qualquer robô móvel para navegar no ambiente em que está inserido e desempenhar uma determinada tarefa, necessita de saber a sua localização. Usualmente estes robôs são dotados de dois tipos de lasers: um no topo do robô, responsável pela localização do mesmo, e outro, denominado laser de segurança, colocado perto do solo com objetivo de evitar obstáculos [7]. Como os lasers de segurança são equipamentos obrigatórios na maioria das aplicações de robôs móveis, porque não utilizar estes lasers para determinar a sua localização? Deste modo permitirá reduzir custos inerentes a estes robôs tornando-os mais atrativos, economicamente, para a indústria.

Em ambientes onde as suas características estão constantemente a ser alteradas, como por exemplo a circulação de pessoas, a alteração da disposição da mobília ou até mesmo as portas presentes, a complexidade dos algoritmos aumenta de forma drástica, principalmente se forem usados os lasers de segurança do robô [8]. Isto porque, para além de verem a sua capacidade angular de deteção reduzida de 360° para 190° ou 240°, como estes estão colocados próximo do solo, detetam imenso ruído [7]. Assim sendo a comunidade cientifica tem como presente objetivo apresentar soluções mais robustas, eficientes e com resultados fidedignos.

Atendendo às necessidades do mercado atual, é imprescindível o desenvolvimento de plataformas robóticas versáteis que tenham a capacidade de desenvolver mais do que uma tarefa e com uma elevada rentabilidade. Isto deve-se, tal como referido anteriormente, ao facto de os robôs estarem associados a uma só tarefa e ainda a períodos de inatividade devido à necessidade de carga das baterias.

1.3 Objetivos

Com o intuito de responder às questões abordadas anteriormente, a presente dissertação tem como objetivos principais dotar uma base robótica de um algoritmo de localização e navegação eficaz e robusto produzindo assim resultados confiáveis, recorrendo exclusivamente ao laser de segurança implementado no robô. Pretende-se ainda munir o robô com um mecanismo de troca de baterias automáticas sem intervenção humana.

Como objetivo adicional pretende-se a implementação de algoritmos de localização baseados em contornos, ou seja, algoritmos que recorram, exclusivamente, às características do ambiente para se localizar.

Capítulo 2

Revisão Bibliográfica

2.1 Localização de robôs móveis

Para que a navegação e a execução de tarefas associadas a robôs móveis seja efetuada com sucesso (*e.g,* troca ou recarregamento de baterias), que é uma das competências mais desafiantes associadas a este tipo de robôs, é necessário que este tenha uma informação precisa da sua localização, isto é, o robô deve saber a sua localização no meio ambiente [9]. Assim sendo, a localização de robôs móveis tem recebido uma enorme atenção pela comunidade robótica nas últimas décadas [7].

Na literatura é possível encontrar diversos tipos de localização que podem ser divididas em dois grandes tipos: localização/deslocamento relativo e localização absoluta [10, 11]. No entanto como o objetivo desta dissertação é a localização global baseada apenas em características naturais do ambiente, o que por outras palavras significa a não inserção de marcadores artificiais, o foco desta análise será na localização absoluta, mais precisamente a localização probabilística e a localização por ajuste de mapas uma vez que permitem dar resposta a este tipo de requisito. Contudo, como estes dois tipos de localização necessitam de um mecanismo de localização/deslocamento relativo (quer para fusão de dados, quer para estimar a localização do robô mesmo quando não existe informação fidedigna obtida pelos sensores) [12] este tipo de localização também será abordado.

2.1.1 Localização relativa

A localização relativa é o método mais utilizado na localização de robôs móveis devido à sua simplicidade de implementação e à sua acessibilidade [10]. O objetivo principal deste método é a estimação da posição e localização de um robô [10]. Para tal, o método de localização relativa necessita, primeiramente, da posição inicial do robô, para ir atualizando essa mesma posição através da integração dos dados fornecidos pelos sensores presentes no mesmo [11]. Este método, também caracterizado pela sua elevada frequência de atualização da localização, é comummente utilizado para fundir os seus dados com os dados resultantes de técnicas de localização mais complexas (*e.g,* técnicas de localização probabilísticas ou por ajuste de mapas) [12]. No entanto,

ao realizar a integração de dados dos sensores, este método está sujeito à acumulação de erros à medida que o robô se desloca no meio ambiente (erro este proporcional à distância percorrida) [11].

A técnica mais utilizada para este tipo de estimativa é a odometria [11, 12]. Esta técnica recorre à monitorização da rotação de cada uma das rodas do robô (através de *encoders*) com o intuito de determinar o deslocamento das mesmas, e através das equações da cinemática do mesmo, atualiza a sua posição e orientação [10, 13]. São diversas as fontes de erro associadas à odometria, podendo ser divididas em dois tipos: fontes de erro sistemáticas (*e.g,* diferentes diâmetros das rodas do robô, distância real entre rodas,..) e fontes de erro não sistemáticas (*e.g,* derrapagem ou patinagem das rodas do robô) [10], tornando este método cada vez mais impreciso à medida que a distância percorrida pelo robô aumenta, inviabilizando o mesmo para grandes distâncias.

Outra técnica utilizada para a localização relativa, é a utilização de um sensor denominado IMU (*Inertial measurement unit*) [14]. Este sensor, que é constituído por um acelerómetro (responsável pela medição da aceleração linear do veículo através da segunda lei de Newton - $F = ma$), um giroscópio (instrumento capaz de medir a velocidade angular segundo um eixo) e um magnetómetro (cuja função é a medição do campo magnético com intuito de obter a orientação do robô), tem a capacidade de estimar a posição e orientação do robô através da integração da aceleração e velocidade angular através de um processamento dos dados obtidos pelos sensores constituintes [14]. Caso este sensor seja utilizado em ambientes *indoor*, o funcionamento do magnetómetro é fortemente influenciado devido ao ruído eletromagnético gerado neste tipo de ambiente, provocando interferências na leitura do campo magnético [14]. Já os acelerómetros e os giroscópios mais acessíveis (*i.e, low cost*) possuem *offsets* consideráveis nas suas medidas, necessitando de um processo exaustivo de calibração [15]. Contudo, apesar dos problemas de precisão inerente às IMUs, esta técnica de localização relativa é frequentemente utilizada em conjunto com a odometria, de modo a validar os resultados da mesma (*e.g,* em casos de derrapagem ou patinagem das rodas do robô) tornando este método de localização mais preciso [14].

2.1.2 Localização probabilística

Inerente ao movimento do robô e às medidas obtidas pelos sensores do mesmo, está associada uma incerteza devido ao ruído presente nos dados adquiridos. Os algoritmos de localização probabilística têm a capacidade de representar essa mesma incerteza, uma vez que, através de distribuições de probabilidade, representam o grau de confiança dos dados que estão a ser processados. Por outras palavras, a localização probabilística estima, para cada instante, a distribuição de probabilidade associada à localização do robô no ambiente, utilizando os dados medidos pelos sensores e a distribuição atual de probabilidade do estado do robô. Assim sendo, para que seja possível estimar a localização (recorrendo a este método) é necessário que o algoritmo tenha acesso à distribuição inicial da probabilidade da sua localização, ao modelo probabilístico (quer dos sensores, quer do movimento do robô) aos dados provenientes dos sensores e ainda a um mapa que representa o ambiente circundante do robô.

Na literatura é possível encontrar diversas abordagens que se enquadram neste tipo de localização, nomeadamente, Filtro de Kalman, Filtros de Kalman multi-hipótese, Localização de Markov e ainda o algoritmo de Localização Monte Carlo e suas evoluções [8]. Os algoritmos baseados no Filtro de Kalman necessitam que o modelo probabilístico dos sensores e do movimento do robô sejam Gaussianos e com média nula, o que leva a uma perda de informação obtida pelos sensores quando este não coincide com o modelo dos sensores/movimento do robô [16, 17, 18]. Para além deste facto, este tipo de algoritmos não têm a capacidade de recuperar de situações de ambiguidade ou "rapto" do robô, ou seja, quando este está completamente perdido [16, 17, 18]. Contrariamente aos Filtros de Kalman, algoritmos baseados na Localização de Markov e na Localização Monte Carlo, aceitam diferentes modelos probabilísticos e permitem recuperar de situações onde o robô se encontra totalmente perdido, no entanto apresentam a desvantagem de serem mais pesados computacionalmente [8].

Consequentemente, as secções seguintes apenas irão incidir nos últimos dois algoritmos referidos, isto é, o algoritmo de localização de Markov e o algoritmo de localização Monte Carlo.

2.1.2.1 Localização de Markov

O método de localização de Markov permitiu resolver os problemas presentes nos filtros de Kalman, ou seja, colmatou o facto de estes não suportarem diferentes distribuições de probabilidade para além das distribuições Gaussianas e ainda de não terem a capacidade de recuperar de situações ambíguas e de "rapto" do robô [19].

Este método é um algoritmo probabilístico que deriva diretamente do filtro de Bayes. O princípio primordial deste filtro passa por estimar, de forma recursiva, a distribuição de probabilidade recorrendo ao estado atual do robô e às medidas obtidas pelos sensores até ao momento [8]. Portanto, o objetivo do filtro de Bayes é calcular a distribuição de probabilidade no instante t recorrendo às informações adquiridas até ao instante *t-1* juntamente com a informação atual do estado do robô e dos dados obtidos pelos sensores [20]. Apesar do método de localização de Markov ser uma aplicação direta do filtro de Bayes (ou seja, com o mesmo objetivo) no âmbito da localização, este necessita ainda de um mapa, discretizado, do ambiente circundante do robô [8] dado que, em vez de manter hipóteses únicas da possível localização robô, este calcula a distribuição de probabilidade para cada célula do mapa. Assim, cada célula do mapa contém a probabilidade de o robô estar nessa mesma célula com uma determinada orientação [21].

Como é possível verificar na Figura 2.1, a probabilidade associada a cada célula é atualizada sempre que o robô se move, transportando as probabilidades de uma célula para outra de acordo com o modelo de movimento do robô (*e.g*, a cinemática do robô), ou quando uma nova medida é adquirida pelos sensores [19]. Para a primeira situação, o algoritmo modeliza o movimento do robô por uma probabilidade condicional, nomeadamente $p(x_t|x_{t-1},u_t,m)$, ou seja, qual a probabilidade de o robô estar no local x_t sabendo que estava anteriormente no local x_{t-1} e que sofreu uma atuação u_t (a variável m é o mapa do algoritmo). Para a segunda situação, isto é, quando são adquiridos novos dados pelos sensores, o algoritmo recorre novamente a uma probabilidade

1: **Algorithm Markov_localization(**$bel(x_{t-1}), u_t, z_t, m$**):**
2: *for all* x_t *do*
3: $\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1}, m)\ bel(x_{t-1})\ dx$
4: $bel(x_t) = \eta\ p(z_t \mid x_t, m)\ \overline{bel}(x_t)$
5: *endfor*
6: *return* $bel(x_t)$

Figura 2.1: Pseudo-algoritmo da localização de Markov [8]

condicionada para modelizar a nova probabilidade - $p(z_t|x_t,m)$ - que tem como significado a probabilidade de obter a medida z_t estando no local x_t. Contudo, como estes cálculos têm de ser executados para todas as células do mapa em cada iteração, este algoritmo necessita de elevados recursos computacionais para a sua execução [16]. Consequentemente, o tempo de processamento também é afetado, levando a que, por vezes, seja inviável aplicar o algoritmo em aplicações em tempo-real [17]. Para além destas desvantagens, o método de localização de Markov vê a sua precisão limitada pela resolução do mapa presente no robô [16, 17]. No entanto, apesar das suas desvantagens, este é um método poderoso para a resolução de problemas de localização global em ambientes dinâmicos.

2.1.2.2 Localização Monte Carlo

No que diz respeito aos métodos responsáveis pela localização global em ambientes dinâmicos, o método de localização Monte Carlo (MCL), também conhecido como filtro de partículas, é o mais popular na comunidade robótica, tendo revelado bastante sucesso na sua aplicação [8, 22, 23, 24]. Este algoritmo, proposto por Thrun *et al* [16, 17] em 1999, é um algoritmo recursivo, baseado na regra de Bayes, que representa a função densidade de probabilidade por um conjunto de N amostras/partículas (geradas aleatoriamente e de acordo com a probabilidade anterior) com um determinado peso associado, distribuídas ao longo de um mapa [16, 17]. Assim, quanto mais partículas existirem num determinado local (maior concentração de partículas), maior será a probabilidade desse local ser considerado como a verdadeira posição do robô [8, 16, 17]. Inerente a este tipo de representação, estão associadas diversas vantagens nomeadamente, a capacidade de representar as mais diversas distribuições de probabilidade (associadas aos sensores e ao movimento do robô), redução drástica do tempo de processamento e quantidade de memória necessária para a implementação deste algoritmo quando comparado com o algoritmo de localização de Markov, concentração dos recursos computacionais apenas nas áreas onde existe a possibilidade de localização do robô e ainda a facilidade de implementação do algoritmo [16, 17, 18].

Uma descrição do algoritmo básico é apresentada na Figura 2.2 sob a forma de pseudo-código.

1: **Algorithm MCL($\mathcal{X}_{t-1}, u_t, z_t, m$):**
2: $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
3: for $m = 1$ to M do
4: $x_t^{[m]} = \textbf{sample\_motion\_model}(u_t, x_{t-1}^{[m]})$
5: $w_t^{[m]} = \textbf{measurement\_model}(z_t, x_t^{[m]}, m)$
6: $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
7: endfor
8: for $m = 1$ to M do
9: draw i with probability $\propto w_t^{[i]}$
10: add $x_t^{[i]}$ to $\mathcal{X}_t$
11: endfor
12: return $\mathcal{X}_t$

Figura 2.2: Pseudo-código do algoritmo MCL básico [8]

Tal como foi referido anteriormente, este algoritmo calcula o novo conjunto de partículas (X_t) recursivamente a partir do conjunto anterior (X_{t-1}). Para o cálculo do novo conjunto de partículas, o algoritmo tem ainda em conta o modelo de movimentação do robô (u_t), as medidas obtidas pelos sensores (z_t) e ainda o mapa do meio ambiente onde o robô está inserido (m). Analisando o algoritmo é possível constatar que este pode ser decomposto em 3 fases: uma primeira fase de predição, isto é, aplicação do modelo de movimento do robô às partículas anteriores (linha 4), uma segunda fase de perceção onde é calculada, tendo em conta a informação proveniente dos sensores do robô, o peso/importância da localização calculada na fase anterior (linha 5) e uma última de reamostragem onde o novo conjunto de M partículas (X_t) é redesenhado no mapa de acordo com a nova distribuição de probabilidade, isto é, apenas sobrevivem as partículas com maior peso/probabilidade [16, 17, 23]. Deste modo, as partículas vão se concentrando na verdadeira posição do robô.

No entanto, à versão básica do algoritmo MCL estão, também, associadas algumas limitações. Mais detalhadamente, uma das limitações tem haver com a precisão do algoritmo, uma vez que esta varia com o número de partículas utilizadas. Caso o número de partículas utilizadas seja muito pequeno, o algoritmo de localização poderá falhar, por consequência de existir a possibilidade de não serem lançadas partículas na verdadeira posição do robô [8, 18]. Consequentemente, nenhuma partícula sobrevirá e o algoritmo de localização falhará. Por outro lado, a versão básica do algoritmo não está preparada para dar resposta à situação de "rapto" do robô, pois quando a posição do robô está bem localizada, todas as partículas concentram-se perto dessa mesma posição [8]. No caso de o robô ser transportado para outro local ou até mesmo a localização estimada estiver errada, como não existe partículas no mapa para além da posição estimada, o algoritmo torna-se incapaz de de recuperar destas situações [8].

Com o intuito de resolver o problema de "rapto" do robô, Thrun *et al* [8], propuseram o

Augmented MCL (Figura 2.3), cujo o princípio se baseia numa heurística de injeção de partículas aleatórias ao conjunto de partículas criado pelo algoritmo básico. Os autores propuseram duas alternativas de lançamento de novas partículas: ou lançando um número fixo de partículas aleatórias em cada interação, ou lançando um número de partículas de acordo com a estimativa da precisão da localização. Para a estimação da precisão da localização, os autores sugeriram a monitorização da probabilidade das mediadas dos sensores (equação 2.1) e relacionando-a com a probabilidade média das medidas, que pode ser obtida pela média do fator peso (equação 2.2) [8].

$$p(z_t|z_{t-1}, u_t, m) \tag{2.1}$$

$$\frac{1}{M}\sum_{m=1}^{M} w_t^{[m]} \tag{2.2}$$

Desta forma, serão lançadas mais ou menos partículas aleatórias conforme a relação entre as medidas dos sensores e probabilidade média das medidas, tornando assim o algoritmo mais robusto [8].

Algorithm Augmented_MCL($\mathcal{X}_{t-1}, u_t, z_t, m$)**:**
- static $w_{\text{slow}},\ w_{\text{fast}}$
- $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
- for $m = 1$ to M do
  - $x_t^{[m]} =$ **sample_motion_model**$(u_t, x_{t-1}^{[m]})$
  - $w_t^{[m]} =$ **measurement_model**$(z_t, x_t^{[m]}, m)$
  - $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
  - $w_{\text{avg}} = w_{\text{avg}} + \frac{1}{M}\ w_t^{[m]}$
- endfor
- $w_{\text{slow}} = w_{\text{slow}} + \alpha_{\text{slow}}(w_{\text{avg}} - w_{\text{slow}})$
- $w_{\text{fast}} = w_{\text{fast}} + \alpha_{\text{fast}}(w_{\text{avg}} - w_{\text{fast}})$
- for $m = 1$ to M do
  - with probability $\max(0.0,\ 1.0 - w_{\text{fast}}/w_{\text{slow}})$ do
    - add random pose to $\mathcal{X}_t$
  - else
    - draw $i \in \{1, \ldots, N\}$ with probability $\propto w_t^{[i]}$
    - add $x_t^{[i]}$ to $\mathcal{X}_t$
  - endwith
- endfor
- return $\mathcal{X}_t$

Figura 2.3: Pseudo-código do algoritmo *Augmented MCL* [8]

Apesar do algoritmo *Augmented MCL* resolver o problema de rapto do robô este ainda apresenta uma limitação. Caso sejam utilizados sensores muito precisos, a performance do algoritmo

degrada-se uma vez que, com sensores muito precisos, um pequeno desvio nos valores dos sensores provoca uma alteração drástica no peso das partículas, levando a que estas sejam descartadas [18]. Uma solução possível seria a adição de ruído artificial nos dados obtidos pelos sensores [8]. No entanto, uma solução mais robusta foi proposta em [18], onde é proposto um algoritmo denominado por *Mixture-MCL*, onde é combinado o algoritmo MCL básico com o seu dual (dual MCL). O algoritmo dual MCL (Figura 2.4) inverte o processo de amostragem quando comparado com o original [18, 25]. Enquanto que o algoritmo original, na fase de predição, estima a posição do robô (*i.e,* calcula as novas partículas) aplicando o modelo de movimento do robô (*e.g,* odometria) e na fase de perceção, viabiliza as estimativas (*i.e,* atribui pesos/confiança às partículas lançadas na fase anterior) tendo em conta os dados obtidos pelos sensores, o algoritmo dual MCL, inverte este processo, utilizando, na fase de predição, os dados obtidos pelos sensores para a estimativa da localização do robô (linha 4 da Figura 2.4), e em seguida na fase de perceção, recorre ao modelo de movimento do robô para atribuir pesos às partículas geradas na fase anterior (linha 5 da Figura 2.4) [18, 25]. Por outras palavras, o algoritmo dual MCL, primeiramente lança

```
1 N                                          ▷ Total number of samples
2 L_t = {}
3 while (size(L_t) < N) do
4     obtain sample ~ p(l_t|o_t)             ▷ Dual Prediction step
5     filter sample using ~ p(l_t|l_{t-1})   ▷ Dual Filtering step
6     if p(l_t|l_{t-1}) > 0 then
7         include sample to L_t
8     end if
9 end while
```

Figura 2.4: Pseudo-código do algoritmo Dual MCL; L_t é o conjunto de partículas; o_t são as medidas realizadas pelos sensores; l_t e l_{t-1} são as variáveis que representam os locais estimados [25]

partículas em zonas do mapa onde estas se aproximem das medidas obtidos pelos sensores (o_t), verificando de seguida qual a probabilidade dessas mesmas partículas (*i.e,* atribui pesos às mesmas) tendo em conta o conjunto anterior de partículas e o modelo de movimento do robô [18, 25]. O *Mixture-MCL* (Figura 2.5) proposto por Thrun *et al* [18] combina estes dois algoritmos, isto é, combina as amostras geradas pelo algoritmo original MCL com as amostras geradas pelo dual MCL, recorrendo a um rácio de mistura ϕ (probabilístico, ou seja, $0 < \phi < 1$) [18, 25]. Deste modo, o método de amostragem de novas partículas do algoritmo original é utilizado com uma probabilidade de $1 - \phi$, enquanto que o processo de amostragem inerente ao algoritmo dual MCL, é utilizado com uma probabilidade ϕ [18, 25]. Com a implementação deste método, Thrun *et al* [18] demonstraram que o algoritmo era robusto o suficiente, não só para lidar com sensores em que o nível de ruído é baixo, como também se conseguirá localizar mais rapidamente em caso de perda de posição do robô (rapto do robô) ou até mesmo com um número significativamente baixo de partículas (*e.g*, 50 partículas). No entanto apresenta como desvantagem principal, a necessidade de modelos de sensores que permitam uma amostragem rápida das posições [18].

```
1   N                                          ▷ Total number of samples
2   L_t = {}
3   while (size(L_t) < N) do
4       obtain r from ~ u(0, 1)
5       if r > φ then
6           obtain sample ~ p(l_t|l_{t-1})   ▷ MCL Prediction step
7           filter sample using ~ p(l_t|o_t) ▷ MCL Filtering step
8           if p(l_t|o_t) > 0 then
9               add sample to L_t
10          end if
11      else
12          obtain sample ~ p(l_t|o_t)       ▷ Dual Prediction step
13          filter sample using ~ p(l_t|l_{t-1})     ▷ Dual Filtering
14  step
15          if p(l_t|l_{t-1}) > 0 then
16              add sample to L_t
17          end if
18      end if
19  end while
```

Figura 2.5: Pseudo-código do algoritmo Mixture MCL [25]

Com a finalidade de determinar o número de partículas necessárias para a localização de um robô, e assim tornar o algoritmo MCL adaptativo, Fox [20] propôs o algoritmo KLD-MCL, cuja a inovação consiste no estabelecimento de limites do erro introduzidos pelas amostras, limitando assim o número de amostras lançadas. Esta abordagem, tem como objetivo principal, lançar o menor número de partículas, isto é, apenas são lançadas as partículas necessárias para que seja possível localizar o robô. Quando o robô está bem localizado, o algoritmo lança menos partículas em relação ao caso em que o robô se encontra mal localizado. Com o lançamento de partículas de forma adaptativa, este algoritmo tem uma complexidade computacional reduzida face às variantes do algoritmo MCL apresentadas até ao momento [20].

Por sua vez, Li *et al* [24] propõem uma nova variante do algoritmo MCL denominada *Merge-MCL*, tendo como principal objetivo o aumento da eficiência computacional do algoritmo MCL. Nesse sentido, os autores propõem a utilização de uma técnica de fusão e divisão de partículas com a finalidade de adaptar o número de partículas dos novos conjuntos. Com a utilização desta técnica, antes da etapa de cálculo do peso, é possível reduzir o número de partículas que o algoritmo tem de atualizar tornando o algoritmo mais eficaz ao nível computacional [24]. Ao utilizar este tipo de método, é tido em consideração não só o peso das partículas, como também a sua similaridade espacial, ao mesmo tempo que é mantida a precisão do algoritmo [24].

Outra vertente do algoritmo, o SAMCL (*Self-adaptative Monte Carlo Localization*), é proposta por Zapata *et al* [23]. Esta nova versão propõe a utilização de duas técnicas distintas: a utilização de uma técnica de *offline pre-catching* (com o intuito de reduzir os custos computacionais, em modo *online*) e uma técnica denominada SER (*Similar Energy Region*) (de modo a responder a situações de "rapto" do robô) [23]. Assim, com a utilização destas duas técnicas, os autores

demonstraram que a versão proposta é computacionalmente mais atrativa em termos de eficiência computacional (quando comparada com o algoritmo MCL original) sendo também uma solução robusta, uma vez que consegue recuperar de situações de "rapto" do robô [23]. No entanto, e tal como todos os outros algoritmos referidos até então, o SAMCL ainda apresenta problemas em estimar a orientação do robô [22]. Com a intenção de melhorar este algoritmo nesta vertente, Hsu *et al* [22] propõem um novo algoritmo, o IMCLROE (*Improved Monte Carlo Localization with Robust Orientation Estimation*). Este algoritmo (que é baseado no SAMCL) incorpora um mecanismo de cálculo da orientação do robô, garantindo assim que cada partícula tenha uma orientação similar à orientação do robô aumentando por sua vez a precisão do algoritmo, uma vez que a posição do robô torna-se mais fácil de estimar [22].

2.1.3 Localização por ajuste de mapas

De acordo com a literatura, este tipo de localização é um método que utiliza um mapa que representa o ambiente onde um robô se inserirá e ainda a estimação da sua posição. Para que a sua posição no mapa seja calculada, é feita uma associação entre este e os dados provenientes dos sensores do robô (posição estimada) . Algoritmos como o Perfect Match e ICP (Iterative Closest Point) permitem realizar esta mesma associação e assim obter a localização global do robô no ambiente [26].

2.1.3.1 Perfect Match

A competição de futebol robótico (Robocup Midsize league) está nas origens do algoritmo Perfect Match, pois para esta competição, a auto-localização associada aos robôs necessita de ser muito fiável, visto ser um requisito essencial para se poder realizar controlo de alto nível como é o caso do planeamento de trajetórias, planeamento de estratégias e até mesmo a coordenação entre múltiplos robôs [27]. Com base nesta necessidade, Lauer *et al.* [27], em 2006, propôs o algoritmo Perfect Match que trata a auto-localização baseando-se apenas nas características naturais do ambiente circundante, isto é, o campo de jogo, nomeadamente as linhas do campo. Como o algoritmo desenvolvido iria integrar um robô dotado com uma câmara omni-direcional presente num campo de jogo, onde as decisões têm de ser tomadas com uma frequência muito elevada (restrições de tempo real), este tinha como objetivo possuir uma auto-localização robusta, precisa e ao mesmo tempo ser eficiente, ou seja, ser leve computacionalmente. Assim sendo, o algoritmo proposto por estes autores, associa o problema de localização a um problema de minimização do erro usando, para tal, um mecanismo eficiente de minimização numérica: o *Resilient Propagation* (RPROP) [28], comummente usado em redes neuronais. Contudo a função de custo utilizada pelos autores difere das funções normalmente utilizadas (quadrado do erro ou o valor absoluto do mesmo) visto não serem robustas o suficiente para lidarem com *outliers* [29]. Por conseguinte os

autores utilizaram a função representada na figura 2.6 sendo modelada pela equação 2.3.

$$E() = 1 - \frac{c^2}{c^2 + (d(p + \begin{bmatrix} cos\phi & sin\phi \\ sin\phi & cos\phi \end{bmatrix} s_i))^2} \tag{2.3}$$

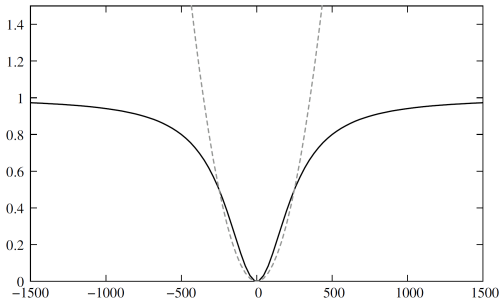


Figura 2.6: Função de custo proposta por Lauer *et al.* [27]

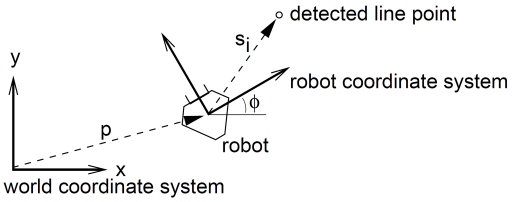


Figura 2.7: Diagrama ilustrativo do sistema de coordenadas global, do sistema de coordenadas relativo do robô e um vetor s_i a apontar para um ponto detetado [27]

Este algoritmo, tem então como objetivo minimizar o erro de *matching* entre os dados adquiridos e o mapa ambiente 2D que o robô possui, mapa este representado por uma grelha em que cada posição contém a distância desse ponto ao objeto mais próximo [27]. É ainda possível, através deste mapa, calcular o mapa de gradiente - em x e em y- que contém dois tipos de informação: informação sobre a direção da variação da distância segundo a direção de x e informação sobre a direção da variação da distância segundo a direção de y [27]. Todos estes mapas podem ser pré-processados, isto é, processados *offline*, para assim aumentar a velocidade de execução do algoritmo Perfect Match aquando a sua aplicação [26]. No que diz respeito à eficiência na utilização de recursos computacionais o algoritmo provou ser bastante leve quando comparado com o filtro de partículas (Tabela 2.1).

Abordagem	Tempo médio de execução por ciclo (ms)
Filtro de partículas com 500 partículas	48.3
Filtro de partículas com 200 partículas	17.9
Perfect Match	4.2

Tabela 2.1: Tempo de processamento: Perfect Match *vs* Filtro de Partículas [27]

Apesar deste algoritmo ter sido desenvolvido com o intuito de satisfazer as necessidades da auto-localização no futebol robótico, Gouveia *et al.* [30], em 2009, demonstraram que o Perfect Match era robusto o suficiente para ser aplicado a diferentes tipos de sensores (*Laser Range Finder* e Infra-Vermelhos) e ainda ser usado para outro tipo de aplicações e ambientes. Nomeadamente, estes investigadores, adaptaram o algoritmo desenvolvido por Lauer *et al.* [27] para ambientes *indoor*, ou seja, as características do ambiente utilizadas para servirem de referência no mapa criado pelo robô eram os objetos (fixos) e paredes presentes no meio circundante. Ao contrário do algoritmo original, Gouveia *et al.* [30] propõem que, a cada interação do algoritmo,

sejam lançadas novas hipóteses de localização do robô, no mapa, de modo a que sejam verificadas posições alternativas à posição estimada pelo robô, percorrendo assim todo o mapa em busca de novas localizações. Atendendo à função de custo, é possível verificar se a posição do robô necessita de ser corrigida. Para evitar que a posição do robô esteja constantemente a mudar, foi proposto que, quando é detetado um ponto que minimiza a função de custo face ao ponto atual estimado, este seja guardado para ser comparado nas próximas iterações. Deste modo é garantido que a estimação da nova posição só é atualizada caso a estimativa, em comparação, seja melhor em N interações consecutivas. Gouveia *et al.* [30] salienta ainda a importância da fusão dos dados fornecidos pelo algoritmo com os dados da odometria, uma vez que, esta fornece dados relativos ao deslocamento e ângulo do robô.

Como este algoritmo despertava cada vez mais o interesse da comunidade científica, em 2012, Pinto *et al.* [12], desenvolveram, a partir dos trabalhos anteriormente mencionados, uma vertente do algoritmo com o objetivo de este usar um mapa tridimensional em oposição do mapa bidimensional, propondo também a utilização de um EKF (*Extended Kalman Filter*) como método de fusão entre os dados fornecidos pelo algoritmo e os dados provenientes da odometria.

Apesar das diferentes adaptações do algoritmo apresentadas até ao momento, todas elas necessitavam que a posição inicial do robô fosse passada, como parâmetro, para a inicialização do algoritmo. Com o intuito de solucionar este problema e tornar o algoritmo ainda mais robusto, Pinto *et al.* [31], em 2013, propuseram uma alteração à etapa inicial deste algoritmo baseada numa abordagem multi-hipótese, demonstrando que, para qualquer posição inicial do robô (desconhecida pelo algoritmo), o algoritmo convergia corretamente para a localização verdadeira do robô apenas com alguma iterações.

Em 2015, Sobreira *et al.* [32], propõem uma alteração quer no critério de paragem do algoritmo RPROP, de modo a garantir que o tempo gasto no algoritmo tenha sempre um limite, quer uma modificação na função de custo (*Figura* 2.8), com o intuito de rejeitar com maior eficácia *outliers*, tornando assim o algoritmo ainda mais robusto (*Figura* 2.9).

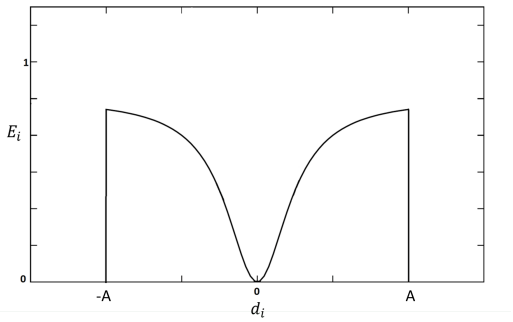


Figura 2.8: Função de custo modificada aplicando um *threshold* (A) [32]

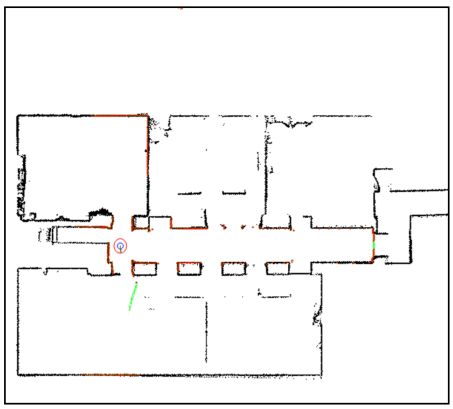


Figura 2.9: Rejeição eficaz de *outliers* utilizando a função de custo representada na Figura 2.8 [32]

2.1.3.2 Iterative Closest Point - ICP

Outro algoritmo muito popular na comunidade científica, utilizado para solucionar problemas de localização por ajuste de mapas, é o algoritmo ICP. Este algoritmo foi inicialmente proposto, em 1992, por Besl *et al.* [33] com o objetivo de responder às necessidades relacionadas com a visão computacional (registo de formas), no entanto foi adotado pela comunidade robótica com o intuito de dar resposta à localização de robôs por ajuste de mapas. Este algoritmo é baseado num processo iterativo, sendo este composto por duas etapas: uma primeira etapa denominada de *matching*, onde é aplicado o critério do ponto mais próximo para estabelecer a correspondência entre entre os pontos do mapa e os dados consecutivos obtidos pelo sensor, e uma segunda etapa de *minimização* cujo objetivo é minimizar o erro de distância (pelo método dos mínimos quadrados) com o intuito de encontrar o deslocamento do sensor. As versões mais comuns deste algoritmo utilizam, para o estabelecimento de correspondências de dados, a distância Euclidiana, distância esta que não diferencia se um ponto está a uma longa distância devido a uma rotação do sensor ou não. Por conseguinte, estas versões do algoritmo não são robustas o suficiente para elevadas rotações do sensor [34].

Com a finalidade de aumentar robustez do algoritmo, Lu *et al.* [35] propuseram a utilização de dois conjuntos de pontos (para a fase de correspondência) em oposição ao algoritmo original, utilizando para um conjunto a distância Euclidiana e para o outro a distância angular. Deste modo, tornaram o algoritmo mais robusto no que diz respeito à rotação do sensor. Todavia, o aumento de robustez implicou um aumento da complexidade computacional, o que por sua vez provocou um aumento do tempo de convergência do algoritmo, visto serem executadas, em cada iteração, as duas etapas do algoritmo em duplicado (uma para o deslocamento e outra para a rotação do sensor). Já Minguez *et al.* [36] propuseram uma nova versão do algoritmo, denominada *Metric-Based ICP* (MbICP), cuja a inovação reside numa nova métrica para a medição da distância tendo em conta a translação e rotação do sensor em simultâneo. Com a implementação desta nova métrica, estes autores conseguiram assegurar que este algoritmo fosse capaz de lidar com grandes erros de odometria (especialmente na rotação) tornando o algoritmo bastante robusto. Para além desta vantagem ainda conseguiram, face às versões do ICP anteriormente referidas, melhorar a precisão, a taxa de convergência e ainda tempo computacional associado ao algoritmo. Armesto *et al.* [34] propuseram a generalização do algoritmo (MbICP) de modo a tornar possível a aplicação deste em ambientes 3D.

Uma outra versão do ICP, o *Point-to-line ICP* (PLICP), proposta por Censi [37], utiliza uma métrica ponto-a-linha em contrapartida à utilizada pelos algoritmos referidos até então (que utilizam uma métrica ponto-a-ponto). Com esta nova métrica, este algoritmo converge de forma quadrática, sendo que supera o algoritmo MbICP em termos de velocidade de convergência, em numero de interações e em precisão. No entanto estes resultados apenas são obtidos caso a posição inicial do robô seja conhecida. Consequentemente, este algoritmo não é suficientemente robusto para lidar com grande erros de odometria ou com grandes rotações do robô (em comparação com o MbICP).

2.1.3.3 Perfect Match *vs* ICP

Tal como foi possível constatar, dois dos algoritmos mais usados para a localização de robôs móveis, utilizando o método de ajuste de mapas, são o *Perfect Match* e o *ICP*. O primeiro, cuja característica principal é a baixa complexidade computacional, tem despertado cada vez mais o interesse da comunidade robótica, enquanto que o segundo é o método comummente usado para este tipo de localização. Com o intuito de comparar este dois métodos de localização, Sobreira *et al.* [26], em 2016, fizeram uma investigação usando várias métricas de comparação, nomeadamente no que concerne à precisão e robustez na presença de *outliers*, peso computacional, velocidade de convergência e erro máximo de inicialização suportado. No que diz respeito aos resultados obtidos, estes autores, demonstraram que relativamente ao peso computacional o algoritmo *Perfect Match* é mais leve (Tabela 2.2).

		Resolução do mapa: 5cm			Resolução do mapa: 1cm		
		288 pontos		1440 pontos	288 pontos	1440 pontos	
	Tempo	*Sem outliers*	*Com outliers*	*Sem outliers*	*Sem outliers*	*Sem outliers*	*Com outliers*
ICP	médio	61	63	274	131	355	478
	máximo	183	189	338	222	545	799
	mínimo	54	55	252	119	332	398
PM	médio	1	1	3	1	5	4
	máximo	2	2	11	3	16	10
	mínimo	<1	<1	2	<1	<1	2

Tabela 2.2: Tempo computacional (em ms) gasto em cada um dos algoritmos mencionados para realizar 50 iterações variando o número de pontos adquiridos pelo laser [26]

.

Quanto à velocidade de convergência, os resultados demonstraram que o algoritmo ICP necessita de menos iterações para convergir quando o robô está a movimentar-se em linha reta ou mesmo parado. Já quando são aplicadas rotações, a performance deste algoritmo degrada-se e o cenário inverte-se. Tal como nas métricas anteriores, em relação ao erro máximo de inicialização tolerado, o *Perfect Match* destaca-se novamente, apresentando uma enorme tolerância quando comparado com ICP (*Figura* 2.10). No que diz respeito à precisão e rejeição de *outliers*, os resultados obtidos pelos autores demonstraram que a performance de ambos os algoritmos é semelhante. Porém, Sobreira *et al.* [26] salienta que apesar de os resultados serem semelhantes, o algoritmo *Perfect Match* apresenta um tempo de execução consideravelmente inferior, o que torna possível um pós-processamento de dados (*e.g,* aplicação de filtros) tornando o algoritmo mais preciso e mais robusto.

2.2 Integração e combinação de algoritmos de localização

Uma localização precisa em robôs móveis é um aspeto fulcral para que este navegue e cumpra as tarefas a ele associadas de uma forma eficaz [9], especialmente se o ambiente onde o robô está inserido, for um ambiente cheio de obstáculos e com elevada densidade de população (ambientes

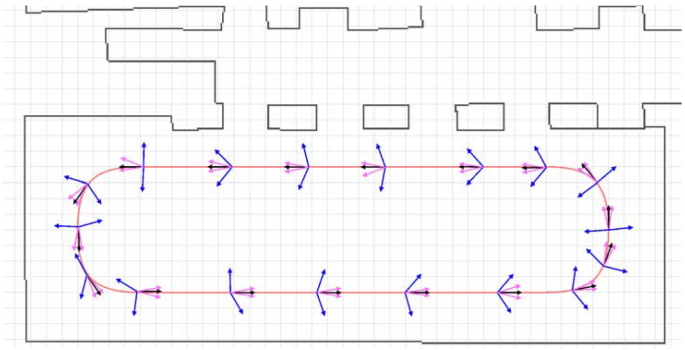


Figura 2.10: Tolerância de rotação associada a cada algoritmo com convergência assegurada. A orientação verdadeira da robô representado a preto, a azul o intervalo de orientação máxima dada pelo *Perfect Match* e a roxo o intervalo de orientação máxima dada pelo algoritmo ICP [26]

dinâmicos) ou se forem utilizados sensores localizados ao nível do solo (aumento da presença de *outliers*) [38].

Em 2016, Vasiljević *et al.* [39] propuseram uma abordagem com a finalidade de aumentar a precisão no sistema de localização de robôs utilizando marcos naturais (isto é, características do meio ambiente) em ambiente industrial. A abordagem proposta, consistia na combinação de três algoritmos, nomeadamente o AMCL (*Adaptative Monte Carlo Localization*), o ICP (*Iterative Closest Point* e um algoritmo de refinamento da solução baseado na DFT (*Discrete Fourier Transform*) aplicados sequencialmente (Figura 2.11) [39]. Como estes algoritmos são aplicados de forma sequencial, ou seja, os resultados de um algoritmo são os dados de entrada do próximo, a estimação da posição do robô vai melhorando em cada etapa, culminando em resultados muito precisos [39]. Mais especificamente, estes autores conseguiram obter resultados com uma precisão de 1cm e 0.5° [39].

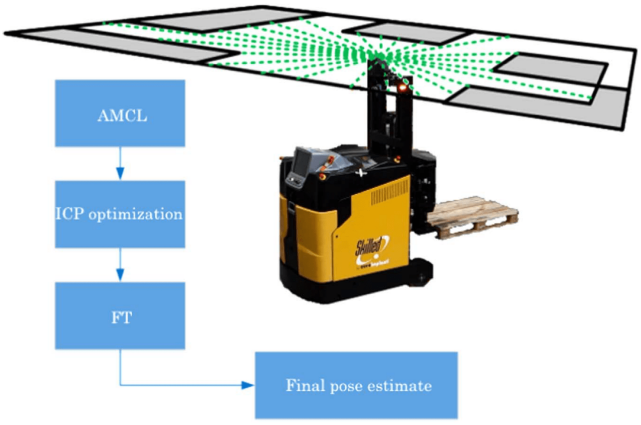


Figura 2.11: Visão geral do mecanismo de localização [32]

Baseado no trabalho Vasiljević *et al.* [39] e com o objetivo de tornar o sistema de localização global mais robusto (isto é, evitar a perda de localização do robô) e preciso, em 2017, Farias *et al.* [38] propuseram um mecanismo de supervisão que consiste na integração de dois (ou mais) algo-

ritmos de localização (para além da odometria do robô) a correrem em paralelo, onde é aplicado um mecanismo de monitorização/supervisão dos mesmos [38]. No seu trabalho, os autores usaram dois métodos de localização global, nomeadamente o AMCL e o *Perfect Match* [38], tendo o primeiro como característica principal a sua enorme robustez e eficácia de localização [38, 39], enquanto que o segundo é caracterizado pela sua precisão de localização, pelo baixo peso computacional e tempo de execução [38, 27]. Portanto, ao serem utilizados mais do que um algoritmo de localização global, torna possível a utilização das melhores características de cada um dos algoritmos [38]. A ideia principal do mecanismo de supervisão proposto, passa pela atribuição de graus de confiança à estimativa de posição fornecida por cada um dos algoritmos de localização (baseados em indicadores de desempenho fornecidos pelos mesmos) e, conforme o grau de confiança atribuído, decidir qual dos algoritmos fornece a estimativa para a posição do robô [38]. No caso em que um dos algoritmos, esteja com indicadores de desempenho relativamente baixos (isto é, com uma baixa precisão da estimativa de localização), quando comparado com o outro, o mecanismo de supervisão utiliza a estimativa dada pelo algoritmo com melhores níveis de desempenho para corrigir a estimativa do algoritmo com baixa precisão [38]. Nas figuras 2.12 e 2.13 está demonstrado o funcionamento do supervisor desenvolvido pelos autores, onde é possível observar a correção das estimativas quando um dos algoritmos começa a perder precisão.

(a) (b) (c)

Figura 2.12: Correção, por parte do mecanismo de supervisão, da estimativa de localização do *Perfect Match* (a azul) baseado na estimativa de localização do algoritmo AMCL (a vermelho) [38]

(a) (b) (c)

Figura 2.13: Correção, por parte do mecanismo de supervisão, da estimativa de localização do AMCL (a vermelho) baseado na estimativa de localização do algoritmo *Perfect Match* (a azul) [38]

No entanto, quando os dois métodos de localização têm os indicadores de precisão semelhantes, o mecanismo de supervisão não realiza nenhuma correção sendo a posição estimada dada pelo *Perfect Match* (visto ser o mais preciso dos dois) [38]. Todavia, se nenhum dos algoritmos apresentar resultados satisfatórios (ou seja, abaixo de uma limiar de desempenho) o algoritmo de localização é suspenso, pois existe probabilidade do robô estar perdido, e é lançado um mecanismo de recuperação com o intuito de relocalizar o robô [38]. Na Figura 2.14 é apresentada, de forma simplificada e sob a forma de diagrama de interação, todo o raciocínio inerente ao mecanismo de supervisão desenvolvido por Farias *et al.* [38].

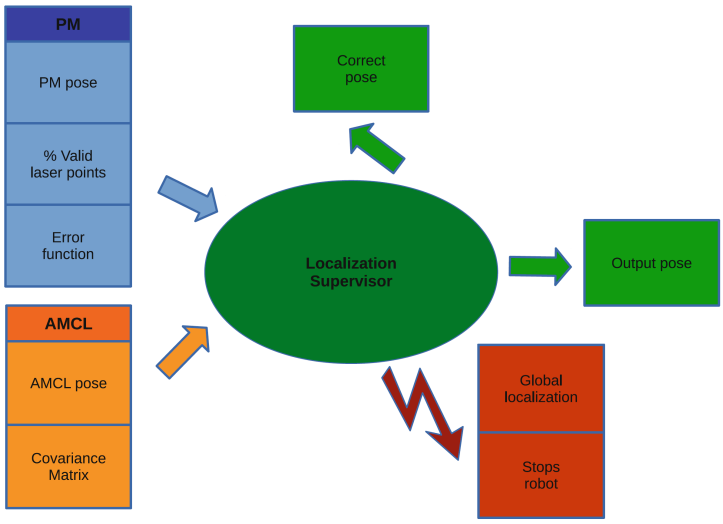


Figura 2.14: Entradas e saídas do algoritmo de supervisão proposto [38]

2.3 Mecanismos de recarregamento e troca automática de baterias

Para que os robôs móveis se movimentem no meio ambiente e executem tarefas a eles atribuídas, estes necessitam de uma fonte de energia para toda a sua componente eletrónica, assim como para os seus motores. Caso haja uma rotura no fornecimento de energia, o robô ficará imobilizado no meio ambiente, tornando-se um equipamento inútil [40, 41, 42]. Portanto, o fornecimento de energia e o modo como este é realizado é um aspeto de elevada importância na comunidade robótica [40, 43, 41, 42], cujo o objetivo principal é a criação de robôs completamente autónomos, ou por outras palavras, a criação de robôs que executem todas as suas tarefas sem a intervenção humana [44].

A solução mais primitiva associada ao fornecimento de energia a robôs móveis, é uma solução denominada por "cordão umbilical" que consiste na ligação direta do robô a uma fonte de energia por meio de um cabo elétrico [40]. No entanto, este tipo de fornecimento de energia limitava, não só a mobilidade do robô (*e.g,* os movimentos de robô tinham de ser controlados pelo facto de existir a possibilidade do cabo ficar preso em obstáculos presentes no meio ambiente), como também o alcance máximo (*i.g,* pelo facto de o cabo ter um comprimento limitado), tornando-se

assim, uma solução inviável [40]. Deste modo, os robôs móveis são comummente dotados de baterias recarregáveis para o fornecimento de energia [40, 41, 42, 44, 45]. Porém, com a utilização de baterias, os robôs vêm a sua autonomia (período de atividade) limitada a um determinado número de horas sendo esta variável conforme a capacidade das baterias [43, 41]. De modo a aumentar a autonomia dos robôs poderiam ser usadas baterias de alta capacidade, no entanto estas são de grandes dimensões e pesos, limitando assim a mobilidade dos robôs [46]. Consequentemente, para que os robôs continuem a realizar as tarefas a eles atribuídas, existe a necessidade de recarregar ou trocar de baterias.

2.3.1 Mecanismos de recarregamento de baterias

Com o intuito de fornecer uma solução para o recarregamento de baterias de forma autónoma (*i.e,* sem intervenção humana), Birk [40] propôs um mecanismo de recarga de baterias básico baseado em contacto direto (Figura 2.16), utilizando para tal a *docking station* (também conhecida como estação de recarga) representada na Figura 2.15. A *docking station* desenvolvida pelo autor, consiste numa placa metálica colocada no solo e uma outra colocada de forma a ficar localizada na zona superior do robô, quando este entra na doca. O robô, dotado com uma antena e com um sistema que o permite estar em contacto com o solo, ao entrar na *docking station* estabelece automaticamente (*i.e,* por contacto direto) uma ligação elétrica com a estrutura da doca, carregando deste modo as suas baterias (Figura 2.16) [40].

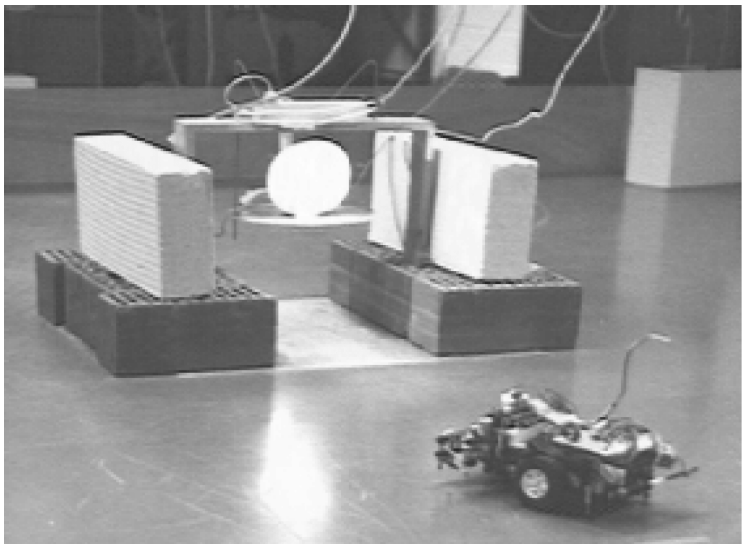


Figura 2.15: *Docking station* desenvolvida [40]

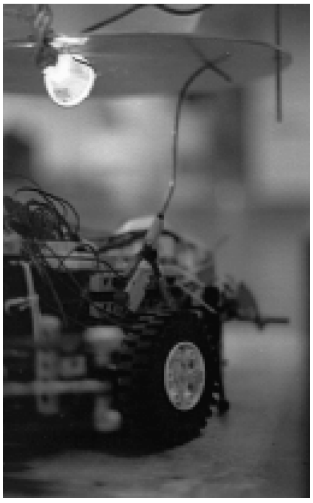


Figura 2.16: Processo de recarga [40]

Já Silverman *et al.* [47, 43], propuseram um mecanismo mais sofisticado para o recarregamento de baterias. Com a mesma finalidade que a solução proposta por Birk [40] (*i.e,* realizar o recarregamento de baterias sem intervenção humana), estes autores desenvolveram um mecanismo de recarregamento de baterias, composto por uma *docking station* com dois graus de liberdade passivos (Figura 2.17) e por um mecanismo de docagem implementado na zona posterior do robô, tendo este apenas um grau de liberdade (Figura 2.18) [47, 43]. Deste modo, os autores tornaram

o sistema de recarga robusto o suficiente para aceitar diferentes ângulos de docagem por parte do robô (60° de tolerância), bem como robôs com diferentes alturas [47, 43]. No que diz respeito ao

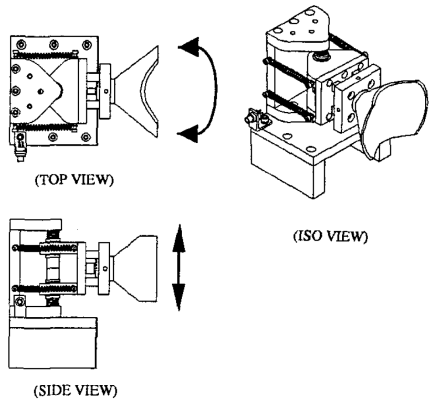


Figura 2.17: *Docking station* modelada (as setas representam a capacidade de movimento da estação) [47]

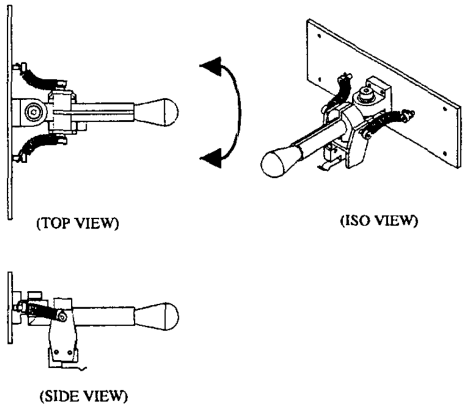


Figura 2.18: Mecanismo de docagem implementado na parte traseira do robô (as setas representam a capacidade de movimento da estação) [47]

processo de docagem, este é realizado quando o nível de bateria atinge um limite mínimo (estipulado pelo utilizador) [47, 43]. Quando este limite é atingido, primeiramente o robô, através de uma câmara, deteta a posição da *docking station*, para seguir na sua direção. Para perceber qual o ângulo de aproximação do robô para com a estação e corrigir assim a sua posição para realizar uma abordagem com sucesso, este recorre a um LRF (*Laser Range Finder*) para detetar o *beacon* implementado por cima da estação de recarga. De modo a indicar ao robô qual estado do processo de docagem, os autores dotaram a estação com um LED de infravermelhos (e o robô com um recetor) que só suspende a emissão quando o processo de docagem foi realizado com sucesso. Caso o processo de docagem falhe, o robô distancia-se da estação e repete todo o processo novamente até obter sucesso. Na Figura 2.19 está representado o sistema desenvolvido pelos autores.

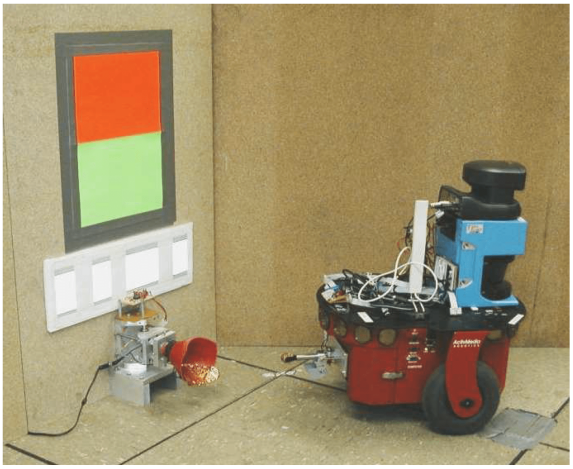
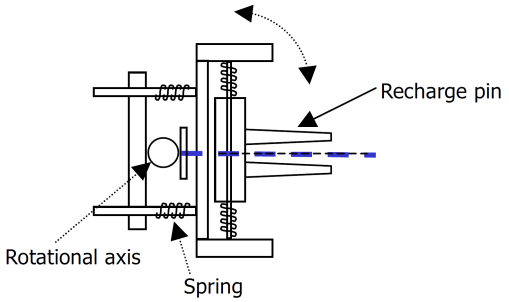


Figura 2.19: Sistema global desenvolvido [43]

Um mecanismo de docagem totalmente passivo e sem graus de liberdade, juntamente com uma *docking station* com dois graus de liberdade passivos foi proposto por Luo *et al.* [41], permitindo ao sistema desenvolvido tolerar pequenos desvios quer no ângulo de abordagem, quer no alinhamento do robô com a estação. Relativamente à estação de recarga desenvolvida por Luo *et al.* [41], esta possui dois pontos de recarga: um para recarregar as baterias associadas aos motores e drivers do robô, e outro para recarregar as baterias responsáveis pelo sistema de sensorização e pelo computador a bordo do robô (responsável pela monitorização, em tempo real, do estado das baterias), bem como um mecanismo (um "braço") para servir de guia na fase final de docagem do robô. Relativamente ao processo de docagem descrito pelos autores, este é baseado inteiramente em visão [41]. Mais detalhadamente, numa fase inicial o robô efetua um movimento de rotação sobre si próprio para detetar um marco artificial colocado na estação de recarga (através da câmara que este detém). Uma vez detetada a *docking station*, é calculada, através da imagem adquirida pelo robô, a distância para a mesma e qual a orientação com que o robô se deve mover para se colocar na frente da estação, para assim finalizar o processo de docagem (Figura 2.21). No entanto,



(a) Undocking condition

Figura 2.20: *Docking station* projetada [41]

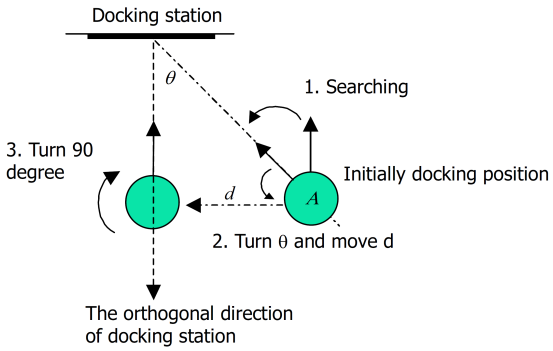


Figura 2.21: Processo de docagem [41]

para além do sistema de recarregamento de baterias proposto, estes autores propuseram ainda um algoritmo de previsão do estado das mesmas (previsão do nível de carga das baterias), para assim tirarem o máximo de partido destas, determinando o momento ideal para as recarregar [47]. Para tal, Luo *et al.* [41] propuseram a utilização de uma técnica de regressão linear para ajustar as curvas de carga e descarga das baterias, simulando assim todo o seu comportamento. O modelo de regressão linear utilizado pelos autores, está representado pela fórmula 2.4, onde y_i é a previsão do estado da bateria na iteração i, β_0 e β_1 são parâmetros de ajustáveis (conforme a bateria em utilização), x_i é o valor do previsor na iteração i, e ε_i é uma variável aleatória com distribuição normal, de média nula ($E\varepsilon_i = 0, i = 1,2,..,n$) e com variância $\sigma\{\varepsilon_i\} = \sigma^2$.

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i \tag{2.4}$$

Com base na *docking station* desenvolvia por Luo *et al.* [41], em [48, 49] são propostas alterações à mesma com intuito de a tornar mais robusta. Nomeadamente, estes autores adicionaram um mecanismo ativo (para além dos mecanismo passivos já implementados no trabalho em que este se baseou) de ajuste horizontal, aumentado assim os graus de liberdade da estação para três

(Figura 2.22) [48, 49]. O processo de deteção da docagem também é ligeiramente diferente, ou seja, é utilizado um LRF (*Laser Range Finder*) para a deteção da estação, em oposição ao método utilizado por Luo *et al.* [41]. Em adição, os autores desenvolveram um módulo de monitorização da energia do robô, com a finalidade de, não só medir a variação da carga da bateria, como também prever qual o tempo de funcionamento do robô até à próxima recarga [48, 49]. Para tal, os autores recorreram a um método de gestão redundante, a um método de estimação de sinal baseado em dados estatísticos e método auto-regressivo [48, 49].

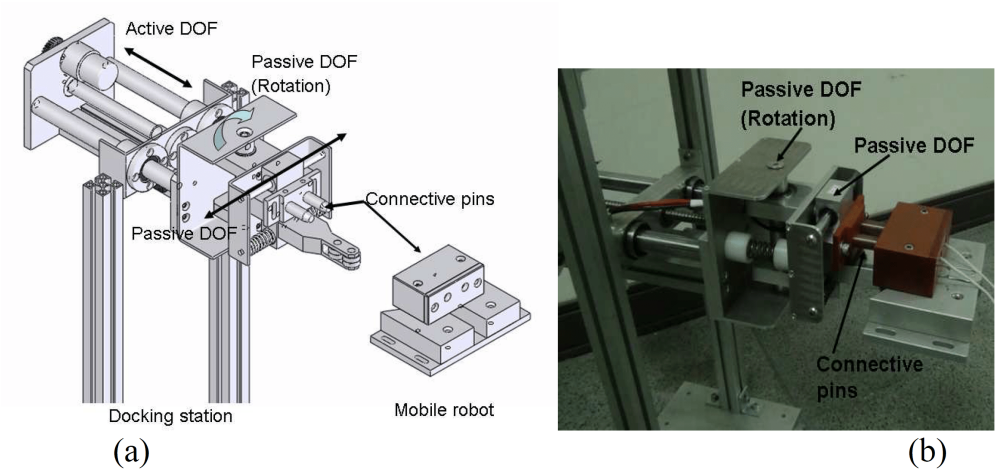


Figura 2.22: Mecanismo de docagem adaptado [48]

Já Cassinis *et al.* [44] propuseram um sistema de recarregamento de baterias *low cost*, todavia eficaz, cuja inovação principal passa pelo modo como é realizada a deteção da estação de recarga. Recorrendo a mecanismos de visão para deteção das *docking stations*, estes autores propuseram a utilização de uma "velha" técnica usada na navegação marítima, mais especificamente utilizando as denominadas *range lights* (*i.e,* luzes de alcance), com o intuito de servir de guia no processo de aproximação e docagem do robô [44].

Com o intuito de satisfazer as necessidades energéticas dos robôs móveis em ambientes domésticos, é proposto por Kim *et al* [50] um mecanismo de recarga de baterias utilizando um robô dotado de sensores infravermelhos (*low cost*) e uma *docking station* passiva onde é utilizada a força do movimento do robô para esta adaptar a sua posição. Desta forma, a estação desenvolvida tem capacidade de compensar os erros de docagem por parte do robô. Mais detalhadamente, a *docking station* desenvolvida por estes autores (Figura 2.23), possui dois graus de liberdade (um rotacional e um linear, de modo a compensar desvios de orientação e posição de docagem, respetivamente), é dotada de LED emissores de infravermelhos, bem como dois braços, com objetivo de acoplar-se ao robô quando este entra na sua zona de ação para assim, ajustar a estação de recarga à posição de entrada do robô, tal como representado na Figura 2.24 [50]. Para que seja possível ao robô realizar o processo de docagem, este necessita de estar numa zona específica (pré-determinada) e dentro do alcance da emissão do LED infravermelhos presentes na estação,

para assim ter uma informação mais precisa acerca da sua localização [50]. Com base na informação obtida pelos sensores, o robô orienta-se e dirige-se para a estação para finalmente realizar o processo de docagem.

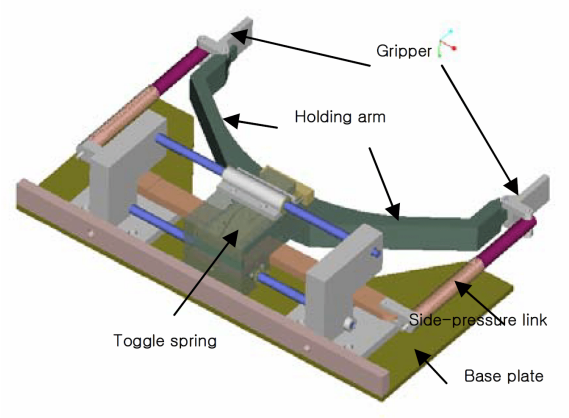


Figura 2.23: Mecanismo de docagem projetado [50]

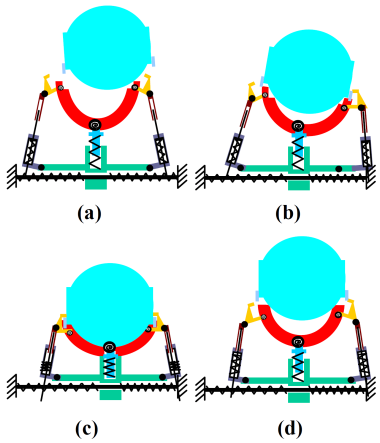


Figura 2.24: Processo de docagem por parte da estação [50]

Tendo como objetivo principal, o desenvolvimento de um sistema de recarga de baterias com uma elevada adaptabilidade, ou por outras palavras, um mecanismo que se adapta a múltiplos robôs com diferentes dimensões, Roh *et al.* [42] propuseram dois tipos de mecanismos, não só com esta propriedade, como também com tolerância de erros de entrada do robô, no processo de docagem. O primeiro sistema proposto pelos autores, diferencia-se dos restantes apresentados até então, pelo facto de possuir três graus de liberdade (Figura 2.25), sendo um deles ativo no sentido vertical, podendo ajustar a altura do braço conforme as dimensões do robô [42]. Com o mecanismo de docagem implementado no robô (Figura 2.26) e com o restantes mecanismos implementados na estação que lhe fornecem dois graus de liberdade (rotativo e linear) tornam esta flexível ao ponto de tolerar erros de entrada na doca, por parte do robô [42]. Já o segundo sistema

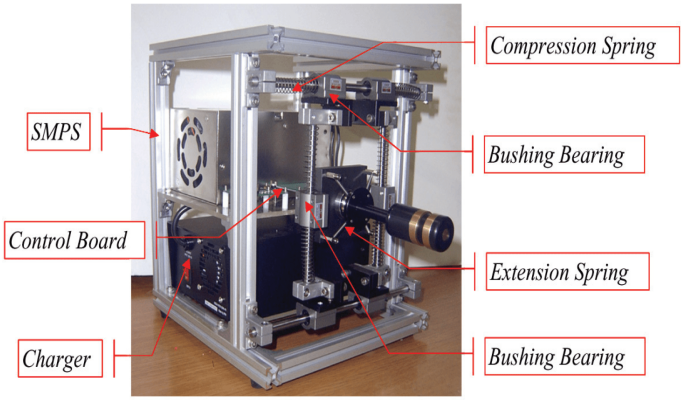


Figura 2.25: *Docking station* projetada [42]

Figura 2.26: Mecanismo de docagem implementado no robô [42]

desenvolvido pelo autores é totalmente passivo [42]. Para conseguirem um sistema de recarga

de baterias totalmente passivo e ao mesmo tempo com os requisitos mencionados anteriormente, os autores recorreram a ímanes. Roh *et al.* [42] recorreram assim, às propriedades das forças magnéticas (*i.e,* atração e repulsão) para tornar o sistema desenvolvido capaz de compensar erros de posição e orientação no processo de docagem. Na Figura 2.27 está representado o conceito utilizado pelos autores no sistema desenvolvido.

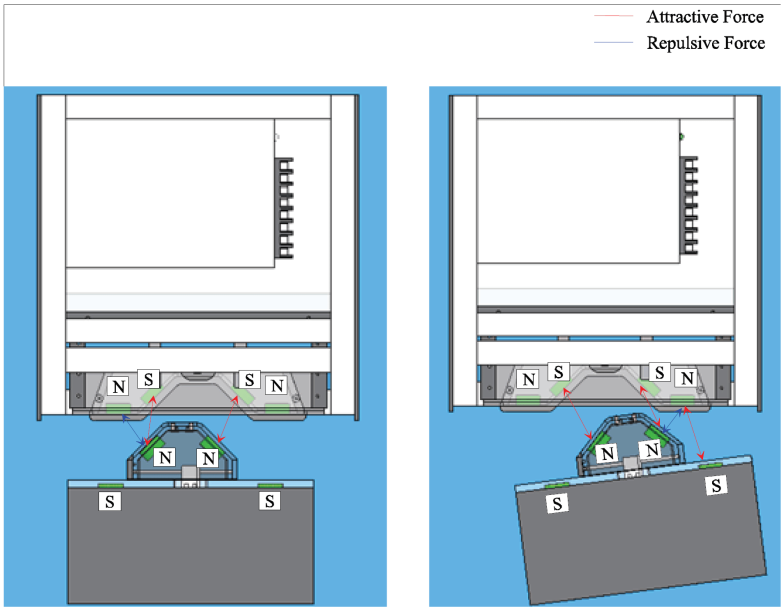


Figura 2.27: Interação entre a *docking station* e o robô através de forças magnéticas [42]

2.3.1.1 Soluções atualmente implementadas no mercado

Atualmente no mercado, o conceito de recarregamento automático de baterias encontra-se aplicado, quer em soluções para ambientes domésticos, quer em soluções para ambientes industriais.

No que concerne aos ambientes domésticos, este tipo de conceito encontra-se implementado em robôs dedicados à limpeza de superfícies, comummente denominados por "Aspiradores Robô". São diversas as soluções existentes no mercado [51, 52, 53, 54, 55], contudo, estas possuem sistemas de recarregamento de baterias mais simples (com o intuito de serem sistemas atrativos/acessíveis em termos económicos), quando comparados com os sistemas apresentados na secção 2.3.1. As *docking stations* inerentes a estes sistemas são relativamente simples, pelo facto de possuírem um carácter totalmente passivo no processo de docagem do robô, isto é, não possuem mecanismos ativos que permitam auxiliar o robô neste processo. No entanto, apesar de não possuírem nenhum mecanismo ativo, estas estações são dotadas de um mecanismo (*e.g,* sistema emissor de infravermelhos - Figura 2.28) que facilita a deteção da mesma, auxiliando assim o robô a deslocar-se na sua direção com uma maior precisão, aumentando a probabilidade de ser efetuada uma docagem com sucesso [51, 52, 53, 54, 55]. No que diz respeito ao processo de

recarregamento, este é realizado, ou por contacto direto do robô com a estação [56], ou por engate [52, 53, 54, 55]. Dado que o objetivo primordial destes robôs passa por um funcionamento

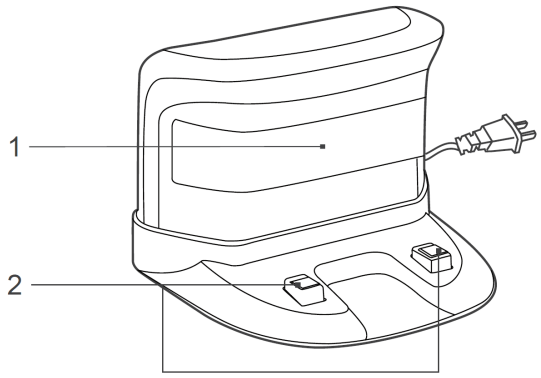


Figura 2.28: *Docking station* proposta pela ECOVACS ROBOTICS; 1- Sistema emissor de infravermelhos; 2 - Mecanismo de engate e recarregamento [57]

completamente autónomo e sem alteração do meio ambiente, os mesmos recorrem a uma técnica de localização e mapeamento em simultâneo (SLAM - *Simultaneous Localization and Mapping*) baseada em características naturais do ambiente, a qual permite a criação do mapa do ambiente circundante do robô (sem intervenção humana), estimando também a sua localização, à medida que este navega no mesmo. Os robôs propostos pela Míele [54], SAMSUNG [52] e iRobot [55], recorrem à visão (*i.e,* recorrem a uma câmara) e aos sensores destinados à deteção de obstáculos (infravermelhos, exceto no caso do robô proposto pela Miele, que recorre a um *Laser Range Finder* (LRF) para este efeito), para realizar o mapeamento do ambiente em que estes estão inseridos. Em oposição à câmara utilizada por estes robôs, o robô proposto pela ECOVACS ROBOTICS [53] recorre a um LRF, enquanto que o robô proposto pela neato [51] recorre a um laser de infravermelhos giratório (ambos, recorrendo igualmente aos sensores de deteção de obstáculos, nomeadamente sensores infravermelhos). No processo de mapeamento, estes robôs procuram pela estação de recarga, registando a sua posição no mapa, permitindo ao mesmo deslocar-se para a localização da *docking station* quando existir a necessidade de recarregar baterias.

O conceito de recarregamento automático de baterias, também se encontra implementado em ambientes industriais. Em termos de soluções para ambiente industrial, destacam-se duas empresas, nomeadamente, a MIR (*Mobile Industrial Robots*) [58] e a OMRON [59]. Ambas as empresas, proporcionam uma plataforma robótica com um recarregamento automático de baterias. Para tal, estas empresas recorrem igualmente a uma estação de recarga passiva, onde o processo de docagem e o método de deteção da estação de recarga é da inteira responsabilidade dos robôs. Apesar de serem estações passivas, estas possuem um formato especial, em forma de "V" (*e.g,* Figura 2.29), para que seja possível a sua identificação por parte do robô, auxiliando este no processo de docagem [58, 59]. De forma similar aos "Aspiradores Robôs", para a deteção da estação de recarga, estes robôs recorrem à técnica de SLAM para mapear todo o ambiente circundante do robô, inclusive a localização da estação de recarga. Desta forma, quando é necessário recarregar

baterias, os robôs consultam o mapa que possuem, com o intuito de saber a localização da estação, deslocando-se posteriormente na sua direção.

Figura 2.29: *Docking station* proposta pela MIR [60]

2.3.2 Mecanismos de troca de baterias

Apesar dos mecanismos de recarregamento de baterias tornarem os robôs móveis totalmente autónomos, ou seja, capazes de realizarem todas as tarefas a eles associadas sem intervenção humana, estes possuem algumas falhas [61, 62, 46, 45]. A este tipo de mecanismos, estão associados períodos de inatividade dos robôs, para estes poderem recarregar as suas baterias, levando a que o tempo gasto neste processo não seja rentabilizado [61, 62, 46, 45]. Muitas aplicações e/ou tarefas (*e.g,* a segurança e vigilância) não toleram este tipo de períodos de inatividade, uma vez que o requisito primordial destas é o funcionamento contínuo 24h por dia [46].

Com o intuito de solucionar o problema referido anteriormente surgiram então os mecanismos de troca de baterias autónomos [61, 62, 46, 45].

De acordo com a literatura, uma das técnicas utilizadas para a troca de baterias, é uma técnica denominada como "sistema de suporte de baterias" [61, 62]. De acordo com [61, 62], esta técnica consiste na utilização de um robô extra, denominado robô de suporte, sendo ele responsável por transportar uma bateria carregada, de modo a efetuar a troca de baterias com o robô em serviço, evitando assim que o último tenha de suspender a tarefa que se encontra a realizar. Saito *et al.* [62], implementaram esta técnica descrevendo uma solução possível (Figura 2.30). A solução proposta pelos autores, utiliza um sistema de controlo, sendo este responsável pela comunicação entre os dois robôs implementados. Assim, o robô de serviço vai informando o sistema de controlo, quer da sua posição, quer do estado da sua bateria, de modo a permitir ao sistema de controlo calcular a necessidade de nova bateria por parte do robô de serviço. Quando esta necessidade é detetada, o sistema de controlo informa o robô de suporte da localização do robô de serviço, dando ordem para transportar uma bateria carregada (obtida na estação de recarga) para o mesmo. Quando o robô de suporte está próximo do robô de serviço, este realiza o processo de docagem (utilizando um sistema de LEDs para orientação), ou seja, é realizado o acoplamento entre os dois robôs, para

efetuar a troca de baterias. Finalizada a troca de baterias o robô de serviço regressa novamente à estação de recarga.

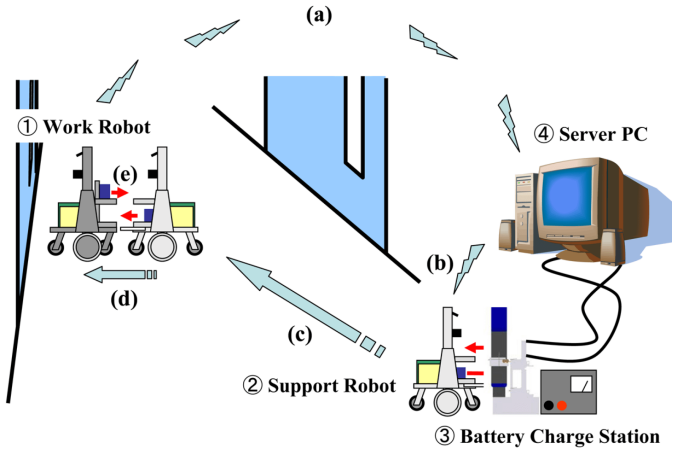


Figura 2.30: Sistema de suporte de baterias proposto [62]

Em contrapartida, Wu *et al.* [46] propuseram uma mecanismo de troca de baterias recorrendo a uma *docking station* (Figura 2.31) evitando a utilização de um robô extra. Para que fosse possível trocar as baterias do robô e ao mesmo tempo manter o mesmo com energia durante todo processo de troca (de modo a manter o robô *online*), os autores dotaram a estação de uma barra equipada com elétrodos que entram em contacto com o robô quando este entra na estação [46]. Para além deste mecanismo, a estação é provida com um sensor ótico (para detetar a presença do robô na estação), de dois braços em forma de gancho (os quais se acoplam ao robô durante o processo de docagem, de modo a ser possível, à estação, adaptar-se à posição do robô, conferindo robustez à estação - Figura 2.32), duas unidades de recarga de baterias, um transportador móvel (com a função de transportar as baterias para/das unidades de recarga), uma fita magnética (com o objetivo de tornar o processo de docagem mais preciso) e de um mecanismo responsável por retirar e colocar as baterias no robô (mecanismo *push/pull*) [46]. Deste modo, quando o robô necessita de trocar baterias dirige-se à estação para realizar o processo de docagem, processo este auxiliado pela fita magnética presente na estação. À medida que o robô efetua o processo de docagem, a estação ajusta-se à posição do robô por intermédio dos braços que esta detém. Quando o processo de docagem é finalizado (detetado pelo sensor ótico), o mecanismo *push/pull* retira a bateria do robô colocando-a sobre o transportador móvel que a encaminha para a unidade de recarga livre, com o intuito de a recarregar. Posteriormente, o transportador móvel dirige-se à outra unidade de recarga para transportar a bateria carregada para o robô, sendo o processo finalizado pelo mecanismo *push/pull* que coloca a bateria dentro do robô. Apesar da complexidade da estação de recarga desenvolvida, o mecanismo proposto por Wu *et al.* [46], é um mecanismo robusto (ou seja, tem a capacidade de compensação de erros no processo de docagem, permitindo erros na posição no processo de docagem por parte do robô) e apenas demora 45 segundos para concluir o processo de troca de baterias, diminuindo consideravelmente o período de inatividade do robô [46].

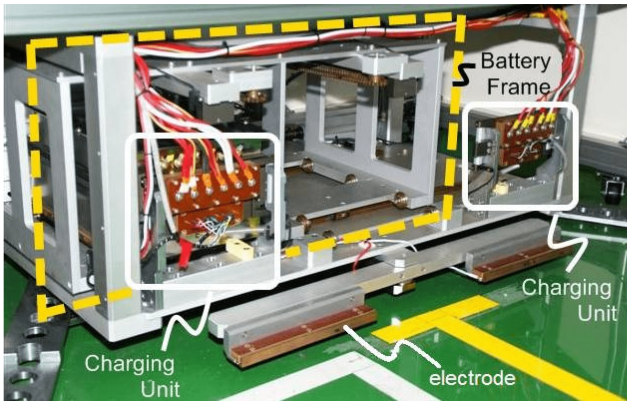


Figura 2.31: *Docking station* projetada [46]

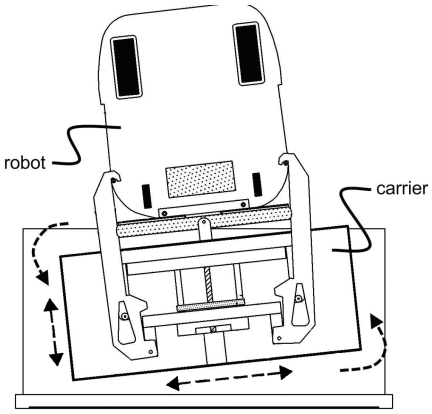


Figura 2.32: Ilustração do mecanismo de ajuste da estação desenvolvida [46]

Numa outra vertente, e com o intuito de solucionar os problemas associados à eficiência de robôs domésticos (*i.e,* o facto de estes necessitarem, ou da intervenção humana para a troca de baterias, ou de necessitarem de longos períodos para recarga de baterias) Wu *et al* [45] propuseram um mecanismo de troca de baterias para este tipo de robôs. O sistema proposto pelos autores, tem por base o desenvolvimento, não só de uma estação de recarga (Figura 2.33), como também um formato do invólucro da bateria (Figura 2.34) [45]. Apesar da diferença considerável em termos dimensionais da estação de recarga, o mecanismo de troca de baterias utilizado pelos autores é semelhante ao mecanismo apresentado (anteriormente) por Wu *et al.* [46]. A diferença principal rege-se à estrutura da estação de recarga (representada na Figura 2.33), nomeadamente ao mecanismo de auxílio ao processo de docagem [45]. Com esta finalidade, os autores dotaram o sistema bloqueio da *docking station* com um disco de sucção eletromagnético, bem como de um sensor de pressão [45]. Desta forma, quando o robô está a realizar o processo de docagem, o disco eletromagnético é ligado, e com recurso à força eletromagnética gerada, o robô é atraído para a estação. Quando o robô entra em contacto com estação, o sensor de pressão ativa-se, indicando assim que o processo de docagem está completo. Concluído o processo de docagem é iniciado o processo de troca de baterias, o qual é muito idêntico ao referido anteriormente (processo demonstrado na Figura 2.35). Com o objetivo de manter o robô *online* (*i.e,* em funcionamento), os autores muniram o robô de baterias de *backup*, sendo estas ativadas (por software) no momento em que o robô inicia o processo de troca de baterias. Uma vez finalizado este processo, o disco eletromagnético é desligado de modo a permitir o desacoplamento do robô.

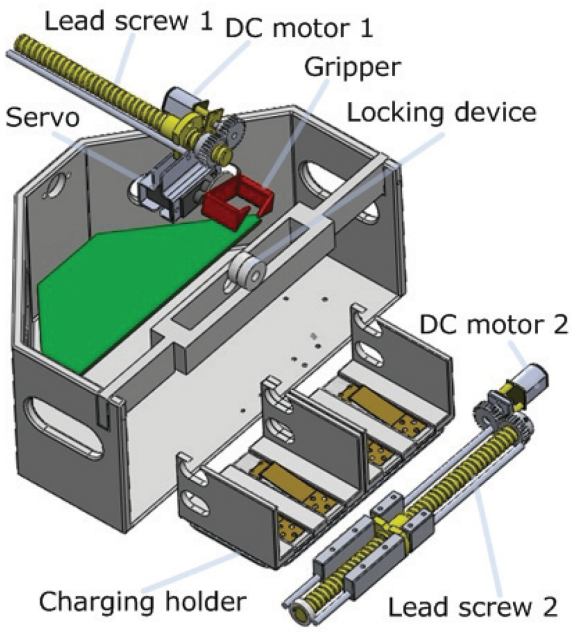


Figura 2.33: Estrutura da estação de recarga [45]

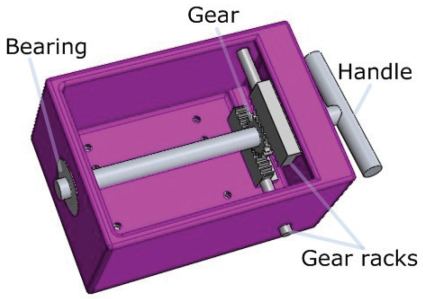


Figura 2.34: Estrutura mecânica do invólucro da bateria [45]

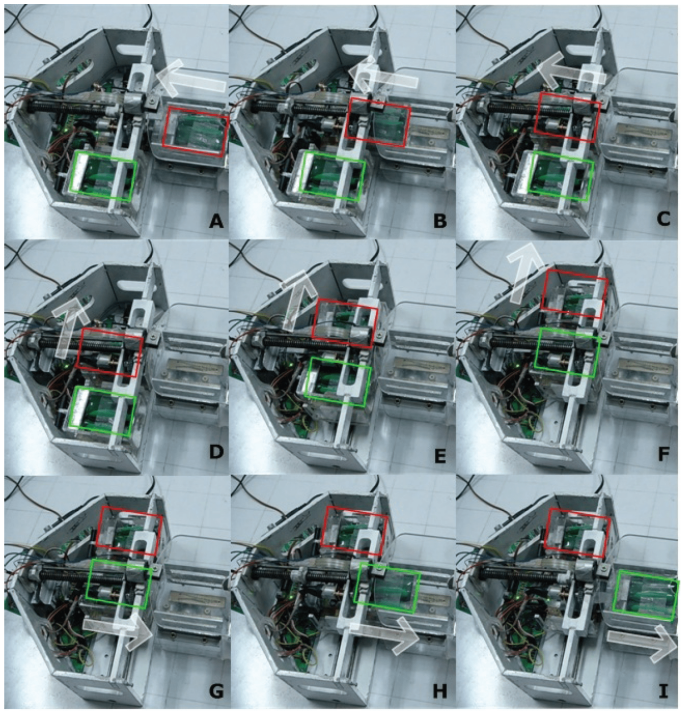


Figura 2.35: Processo de troca de baterias. Assinalado a vermelho a bateria descarrega, e a verde a bateria carregada [45]

Capítulo 3

Caracterização do problema

3.1 Definição do problema

Com a crescente utilização de robôs móveis nas mais diversas aplicações, surge cada vez mais a necessidade de desenvolver robôs versáteis, de alta rentabilidade e com custos comercialmente interessantes.

Com este objetivo, e recorrendo a uma plataforma robótica dotada de um LRF situado próximo do solo e apenas com uma capacidade de deteção de 240°, pretende-se desenvolver um sistema de localização e navegação, em ambientes dinâmicos, robusto e com elevada precisão, utilizando para tal as características naturais (*i.e,* baseada em contornos) do ambiente onde este será inserido.

Por outro lado, como os robôs de pequenas dimensões atualmente presentes no mercado necessitam de suspender as suas atividades para o processo de recarga de baterias, levando a longos períodos de inatividade, pretende-se também desenvolver um mecanismo automático de troca de baterias, permitindo assim eliminar a necessidade de intervenção humana e ainda os períodos de inatividade associados a este tipo de robôs.

3.2 Solução proposta

Com o intuito de responder aos problemas identificados ao nível localização e navegação em ambientes dinâmicos, recorrer-se-à ao desenvolvimento de um supervisor de algoritmos de localização. Mais precisamente, será desenvolvido um supervisor que coordena e supervisiona dois algoritmos de localização, nomeadamente o algoritmo AMCL e o *Perfect Match*, sendo estes executados em paralelo na unidade de processamento do robô.

Já com a finalidade de proporcionar um mecanismo automático de troca de baterias ao robô em questão, será desenvolvida uma estação de recarga (*i.e, docking station*), onde será executada a troca de baterias, ficando a bateria, deixada pelo robô, a recarregar. A *docking station* desenvolvida, será uma estação passiva, isto é, sem elementos ativos na sua constituição, pelo que será o robô a ter o papel ativo neste processo. Para tal o robô, possuirá um mecanismo dedicado à troca

de baterias. De modo a manter o robô *online* e para que este consiga executar todo processo de troca de baterias, este será munido de baterias de *backup*.

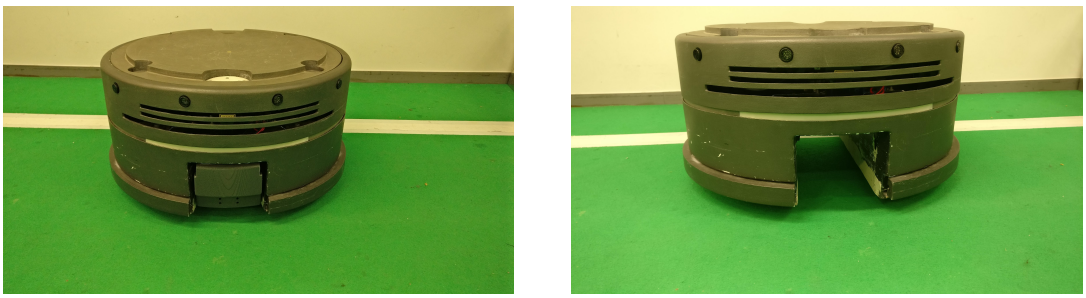
Capítulo 4

Robô Clever

Robô Clever - é assim designado o robô utilizado para o desenvolvimento da presente dissertação. Portanto, nesta secção será realizado uma breve descrição do robô com o intuito de apresentar as características principais do mesmo.

Na figura 4.1 é possível visualizar toda a estrutura física do robô em questão. Este robô (patrocinado pela CLEVERHOUSE), possui uma forma geométrica circular, com um raio de 0.23m e com uma altura de 0.275m. Tal como é possível verificar na figura 4.1, a localização da bateria principal é na zona inferior do robô, o que facilita o processo de troca de baterias (demonstrado no capítulo 6).

No que diz respeito ao modo de locomoção, este utilizada um modo de tração diferencial, encontrando-se as rodas do robô centradas face à forma geométrica do mesmo.



(a) (b)

Figura 4.1: Robô Clever: a) Robô equipado com a bateria principal; b) Robô sem bateria principal.

4.1 Principais componentes eletrónicos

O robô Clever é dotado de vários componentes, desde componentes responsáveis pela alimentação do robô, aos responsáveis pelo processamento de toda a informação adquiridas pelos sensores. Desta forma, neste secção será apresentado os principais componentes eletrónicos presente no robô Clever.

Todos os componentes mencionados nesta secção, já se encontravam presentes no robô antes da elaboração da presente dissertação, pelo que os desenvolvimentos apresentados nesta, apenas residem em termos de desenvolvimento de *software*.

4.1.1 Baterias principais

No que diz respeito à alimentação de todos os componentes eletrónicos presentes no robô, esta é realizada recorrendo à bateria removível (denominada como principal) presente no robô. Tal como referido anteriormente esta situa-se na zona central do robô, na parte inferior. A bateria principal do robô Clever é composta por duas baterias UL 9-12[1] de 12V ligadas em série (fornecendo assim 24V). Na figura 4.2 encontra-se representada a caixa envolvente das duas baterias referidas anteriormente. Tal como é possível verificar esta possui três contactos de cobre (na parte superior e inferior), sendo o contacto central o pino positivo da bateria e os restantes o pino negativo. Desta forma, torna-se possível colocar a bateria no robô em qualquer sentido. Os contactos na zona inferior da bateria permitem que, quando esta seja largada na estação de recarregamento, fiquem imediatamente a recarregar sem ser necessário qualquer intervenção humana.

(a) (b)

Figura 4.2: Bateria do Robô Clever: a) Vista de cima; b) Vista lateral.

4.1.2 Baterias de *backup*

Com o intuito de continuar a fornecer energia ao robô, nomeadamente quando este não possui a bateria principal acoplada (requisito imposto pelo processo de troca de baterias), este encontra-se munido de baterias de *backup* (de pequena dimensão), estando as mesmas representadas na figura 4.3. Tal como é possível observar na figura, de modo a proporcionar os 24V necessários para o funcionamento do robô, são utilizadas quatro baterias UL 4.5-6[2] de 6V ligadas em série. Assim quando o robô larga a bateria principal, este recorre de imediato a estas baterias para poder alimentar todos o componentes eletrónicos presentes no robô, permitindo manter o robô operacional. Adicionalmente, são estas baterias que permitem que o robô se desloque para acoplar a outra bateria

[1] http://ultracell.net/datasheets/UL9-12.pdf
[2] http://ultracell.net/datasheets/UL4.5-6.pdf

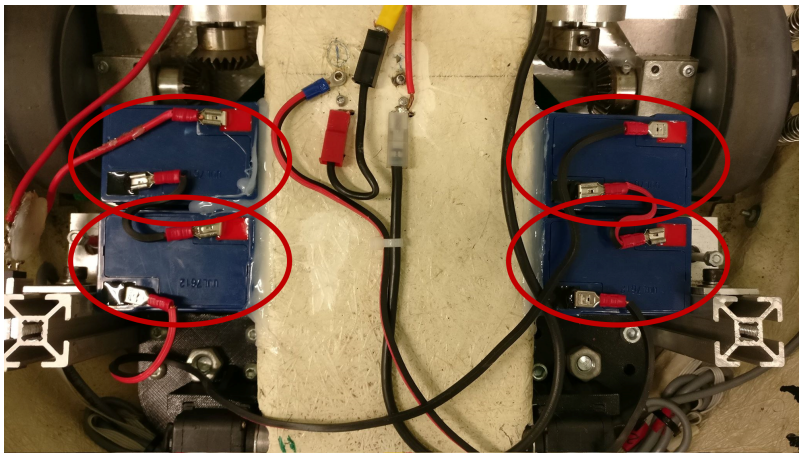
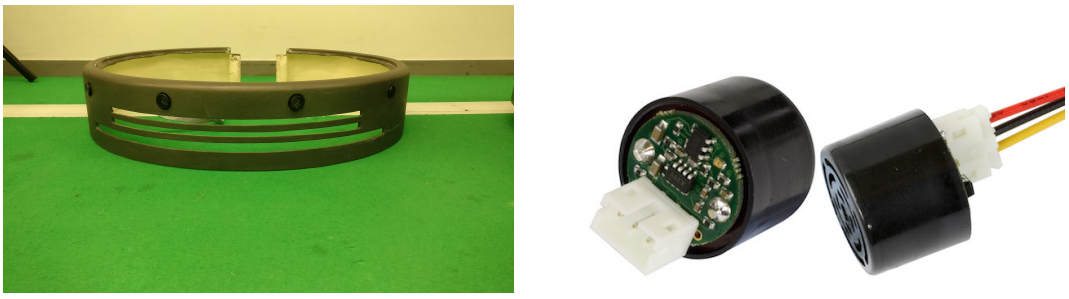


Figura 4.3: Baterias de *backup* do robô Clever.

4.1.3 Sonares e LEDs

O robô Clever, possui na sua zona frontal, quatro sonares SRF01[3] instalados como forma de dotar o robô com um mecanismo complementar para deteção de obstáculos no percurso do robô. Na figura 4.4.a) encontra-se representado a estrutura onde os sonares se encontram instalados. Relativamente aos sonares utilizados, estes encontram-se representados na figura 4.4.b)



(a) (b)

Figura 4.4: a) Sonares presentes na estrutura do robô Clever. b) Sonares SRF01 utilizados

Adicionalmente, o robô utilizado possui ainda um conjunto de LEDs, controlados por *software* na sua zona frontal e traseira, para fins comerciais. Por outras palavras, este sistema é utilizado para atrair a atenção para o robô, especialmente, caso este esteja a ser utilizado para fins publicitários.

[3] https://www.robot-electronics.co.uk/htm/srf01tech.htm

4.1.4 Laser

O laser presente no robô Clever é o laser URG-04LX-UG01[4], estando este montado de forma invertida e na zona superior do robô, tal como é possível observar pela figura 4.5.

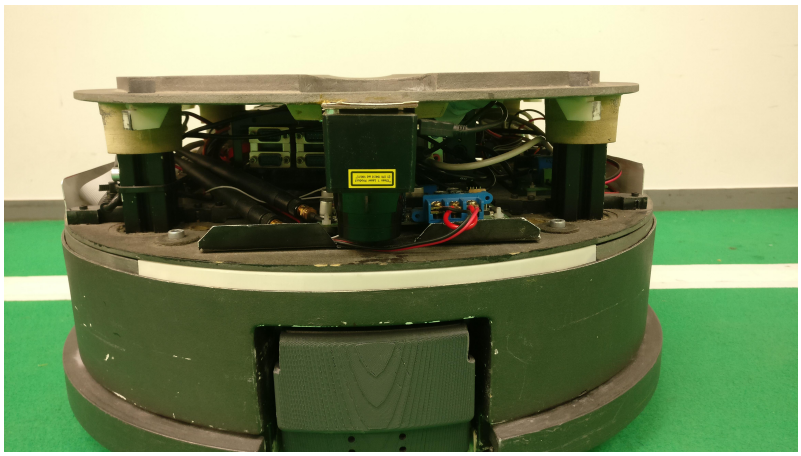


Figura 4.5: Laser (invertido) presente no robô Clever.

No que diz respeito às principais especificações do mesmo, estas encontram-se representadas na tabela.

Tabela 4.1: Principais especificações do laser URG-04LX-UG01.

Tensão de alimentação	5V DC (USB)
Fonte luminosa	Laser infravermelho (λ = 785 nm)
Área de deteção	de 20 a 4000 mm, 240°
Precisão	de 60 a 1000mm: $\pm 30mm$ de 1000 a 4095mm: $\pm 3\%$ da medida obtida
Resolução angular	Incremento do ângulo: 0.36°
Tempo de amostragem	100ms/scan

4.1.5 Servo Motores e mecanismo de encaixe

Dedicado ao mecanismo de troca de baterias, encontram-se instalados quatro servo motores AX-12 da Dynamixel[5]. Estes servos estão diretamente relacionados com o acoplamento da bateria ao robô, sendo estes os responsáveis pelo processo de carga/descarga da bateria. Estes são controlados por *software* recorrendo a uma ligação USB para comunicação com o computador presente no robô.

Na figura 4.6, encontra-se representado os servos utilizados no robô. Já na figura 4.7, encontra-se ilustrada a peça de engate que os servos possuem para o processo de elevação/descida da bateria.

[4] https://www.hokuyo-aut.jp/search/single.php?serial=166

[5] $http://support.robotis.com/en/product/actuator/dynamixel/ax\_series/dxl\_ax\_actuator.htm$

Figura 4.6: Servo motor utilizado no processo de troca de baterias.

Figura 4.7: Peça de alumínio responsável pelo encaixe e acoplamento das baterias principais.

4.1.6 Computador

Para todo o processamento de dados, o robô Clever possui um computador próprio. É neste computador que é executado todo o *software* presente neste mesmo robô. Mais especificamente, o computador possui o sistema operativo Linux, com a distribuição Ubuntu 16.04[6]. Relativamente às características técnicas, as principais encontram-se na tabela 4.2 (estando o mesmo ilustrado na figura 4.8)

Tabela 4.2: Principais características do computador presente no robô Clever

Processador	Intel(R) Core(TM) i5-3340
Frequência	3.10 GHz
Arquitetura	64 bits
Cache	6 MB
RAM	12 GB
ROM	64 GB

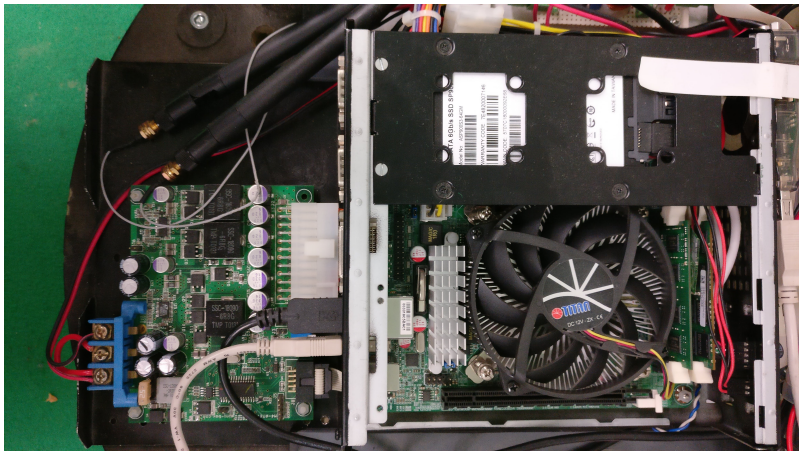


Figura 4.8: Computador presente no robô Clever.

[6]http://www.ubuntu.com/

4.2 *Software*

4.2.1 *Framework* ROS

No que concerne ao *software* presente no robô Clever, este encontra-se desenvolvido tendo em conta a utilização da *framework* ROS[7]. Esta *framework* é dedicada para o desenvolvimento de *software* para robôs. Esta possui uma ampla gama de ferramentas, livrarias e convenções que visam a simplificação da criação de sistemas robóticos complexos para as mais diversas aplicações. Foi desenvolvida também com o intuito de facilitar o trabalho em equipa. Portanto, o sistema possui uma arquitetura modular, permitindo deste modo integrar facilmente diversos módulos distintos. Outra vantagem desta *framework*, deve-se ao facto de esta gerir todas as comunicações entre os diversos módulos, sendo que o utilizador não necessita de se preocupar com as mesmas.

Figura 4.9: Logótipo associado à *framework* utilizada.

4.2.1.1 Simulador *Stage*

Uma das ferramentas que o ROS proporciona é o simulador *Stage*[8]. Este é um simulador 2D que providencia um bom compromisso entre o realismo da simulação e o seu peso computacional. Foi desenvolvido de modo a proporcionar o teste de controladores antes de estes serem aplicados no robô real. No que diz respeito ao simulador, neste é possível simular diferentes robôs móveis (isto é, com diferentes modos de tração: diferencial, triciclo e omnidirecionais) com a respetiva odometria associada. Para além da simulação da odometria do robô, é possível simular ainda o comportamento de inúmeros sensores (como por exemplo, *lasers scanners*). Relativamente à simulação do ambiente, este recorre a um mapa (que contém a disposição as paredes). Para além das paredes estipuladas pelo mapa, é ainda possível simular a presença de objetos externos ao mapa, ou por outras palavras, simular a presença de *outliers*

Na figura 4.10, encontra-se representado o ambiente de simulação utilizado para a aplicação de testes numa fase intermédia do desenvolvimento, ou seja, antes da transição para o robô real.

[7] http://www.ros.org
[8] http://wiki.ros.org/stage

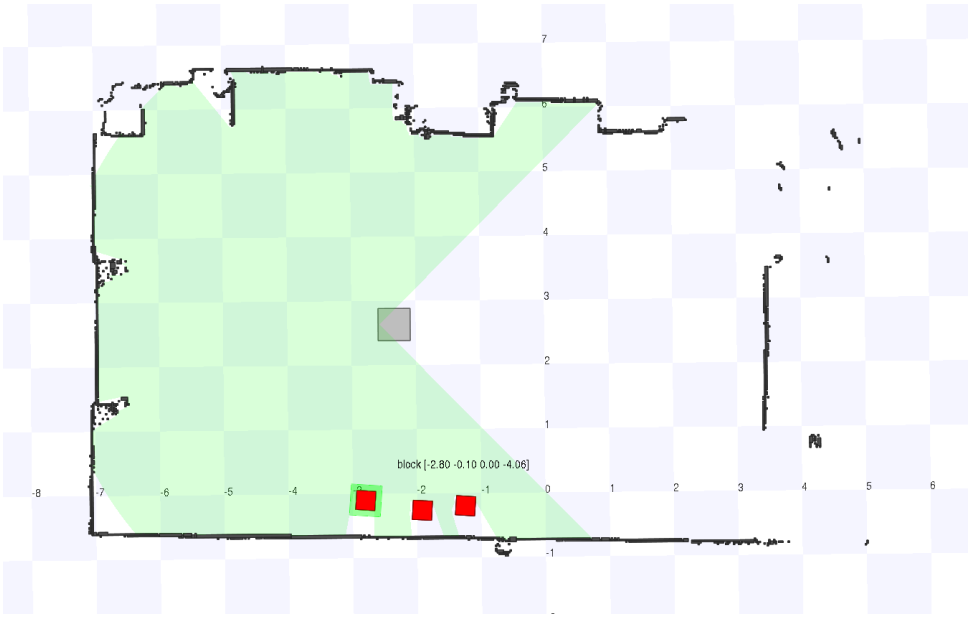


Figura 4.10: Figura ilustrativa do simulador utilizado - *Stage*. A cinzento encontra-se representado o robô Clever. Os objetos com a cor preta são as estruturas fixas do mapa. A vermelho encontra-se ilustrado objetos móveis que permitem a simulação de obstáculos. A região a verde corresponde à área de deteção do laser presente no robô.

Capítulo 5

Supervisor de algoritmos de localização e *tracking* da pose de um robô

Para que um robô navegue autonomamente e com sucesso é necessária uma localização precisa e robusta. Com o intuito de proporcionar e garantir estas características na localização de um robô surgiu o conceito de supervisor. De acordo com a comunidade científica (tal como demonstrado na secção 2.2), um supervisor de localização tem como principal objetivo detetar uma falha nos algoritmos de localização, ou seja, quando estes convergem para uma pose errada (o que provoca uma perda da localização do robô). Deste modo torna-se necessário que os supervisores de algoritmos de localização sejam munidos de mecanismos que permitam detetar possíveis falhas dos algoritmos de localização global e assim agir em conformidade.

De acordo com o estudo precedente a esta dissertação, os algoritmos AMCL e *Perfect Match* provaram ser viáveis e robustos para a sua utilização em robôs móveis em termos de *Pose Tracking* [63]. Portanto, nesta secção, primeiramente, será apresentada uma análise de cada um dos algoritmos com o intuito, não de analisar a sua viabilidade para aplicação em robôs móveis, mas sim detetar situações onde este perdem o *tracking* da posição do robô, ou seja, detetar situações onde existe uma perda de localização. Seguidamente, será analisado o supervisor desenvolvido por Farias *et al.* [38] com a mesma finalidade. Realizadas as análises e detetadas as falhas nos algoritmos de localização e supervisão, será proposto um mecanismo de supervisão com o intuito de solucionar os problemas detetados. Além disso, nesta secção, também será apresentado um estudo de um mecanismo de deteção de falhas dos algoritmos de localização, isto é, um mecanismo que deteta uma situação anómala na localização dada pelos algoritmos de localização global. Por fim é proposta a integração do mecanismo desenvolvido com o supervisor de Farias *et.al.*, com o objetivo de colmatar as lacunas identificadas no algoritmo dos autores. Para todas as análises, testes e desenvolvimentos realizados nesta secção, o robô móvel utilizado será o robô Clever apresentado no capítulo 4.

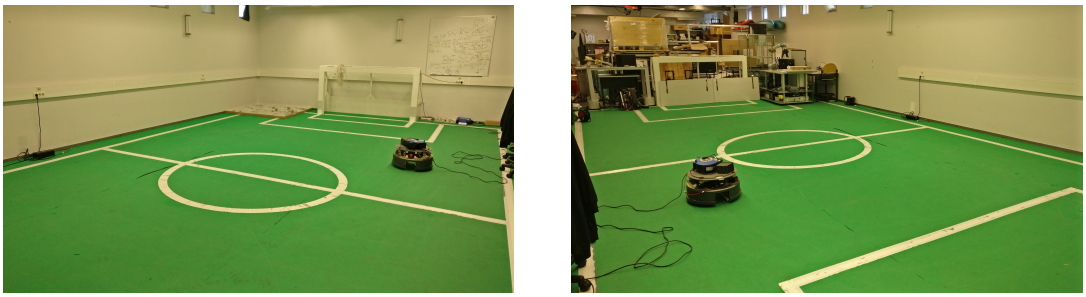
No que diz respeito à análise dos algoritmos, foram desenvolvidos diversos cenários de teste com o intuito de simular as situações mais desfavoráveis que o robô poderá encontrar na sua navegação, para assim procurar situações onde os algoritmos de localização falhassem. Mais

especificamente, os testes desenvolvidos consistiram, em colocar o robô a navegar num ambiente com *outliers*, isto é, a navegar com objetos externos ao mapa. Assim sendo foram introduzidos de forma gradual e diversificada, *outliers* no ambiente de modo a representar as diversas situações, desde a introdução de paredes "falsas" no meio ambiente, à limitação do campo de visão do laser simulando a presença de pessoas em volta do robô. Por fim foram ainda realizados ensaios onde foram forçados erros de odometria com a finalidade de testar a robustez do algoritmo face à má localização relativa. Deste modo foram testados vários cenários de modo a simular a navegação do robô no contexto onde este vai ser aplicado.

Para que um robô se localize e navegue autonomamente no meio ambiente, este necessita de um mapa do meio onde irá navegar. Este mapa deve apenas conter as estruturas e objetos invariantes no tempo, isto é, estruturas e objetos fixos. No presente caso e para aplicação dos testes em específico, o ambiente utilizado encontra-se representado na figura 5.1. Portanto, o ambiente de testes foi mapeado através de um algoritmo disponível na *framework* ROS, nomeadamente o algoritmo *hector slam*[1], resultando o mapa presente na figura 5.2.

Tal como referido anteriormente, a análise efetuada a cada um dos algoritmos mencionados tem como objetivo detetar falhas nos mesmos. Como os algoritmos de localização estimam a pose do robô no referencial global (*i.e,* mapa), é necessário uma referência de localização neste mesmo referencial, para assim se poder comparar a localização do robô com a estimada pelos algoritmos. Como não foi possível a utilização de um *ground truth*, foi calibrada uma trajetória de referência, utilizando pontos específicos no mapa que permitissem verificar visualmente que o robô se encontra bem localizado, ou melhor, que este não possui um erro elevado na sua localização. Assim, executando a mesma trajetória é possível estipular uma localização de referencia para essa mesma trajetória. Portanto, colocou-se o robô a executar uma trajetória (sob a forma de quadrado) num ambiente sem *outliers* (figura 5.1), verificando a localização do mesmo ao longo do seu trajeto, estando a mesma registada na figura 5.3. Para a execução da trajetória, foi utilizado um controlador $GoToXYTheta$. Este controlador apenas tem em conta o ponto de destino, encaminhando o robô para esse mesmo ponto. Ou seja, se o robô partir de diferentes posições iniciais, serão executadas trajetórias diferentes, ao contrário de um controlador $FollowLine$. Por conseguinte, partindo do mesmo ponto, o robô executará sempre a mesma trajetória, ou seja, a estimação da localização do robô será sempre a mesma. Desta forma, caso haja um desvio da localização pretendida, é sinónimo de falha no algoritmo de localização e não do controlador utilizado, permitindo assim verificar falhas no mesmo.

[1] $http://wiki.ros.org/hector\_slam$



(a) (b)

Figura 5.1: Ambiente utilizado para a elaboração do mapa utilizado pelos algoritmos de localização

Figura 5.2: Mapa resultante do algoritmo *hector slam*

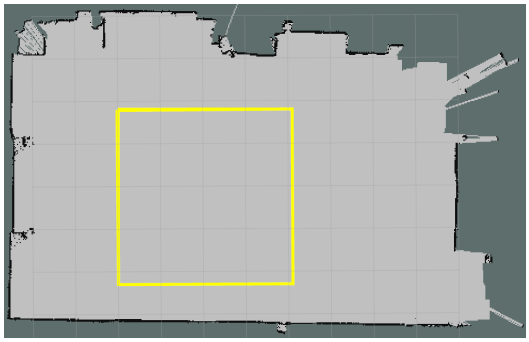


Figura 5.3: Localização esperada do robô (a amarelo) nos testes realizados, obtida através dos testes de calibração de trajetória realizados.

5.1 Princípios teóricos

5.1.1 Referenciais

Um referencial é utilizado para expressar, de forma única, a posição de um ponto no universo. Contudo, o mesmo ponto no universo pode ter coordenadas diferentes caso este seja expresso em referenciais diferentes. Na robótica, é comum utilizarem-se diferentes referenciais uma vez que existem diversos "pontos de vista", isto é, as coordenadas da posição de um objeto depende de onde este está a ser observado. Tal como demonstra a Figura 5.4 a posição (em termos de coordenadas) do objeto relativamente ao Robô 1 é diferente quando observado a partir do Robô 2, todavia o objeto é o mesmo e no referencial do mundo o objeto tem apenas um par de coordenadas.

Nos sistemas de localização de robôs, são frequentemente usados diversos referenciais cartesianos 2D/3D, que podem ou não ser alterados com o decorrer do tempo. Apesar de num sistema robótico existirem múltiplos referenciais, todos eles podem ser convertidos num (qualquer) outro referencial. Para tal é necessário aplicar uma transformação, que é composta por uma rotação e

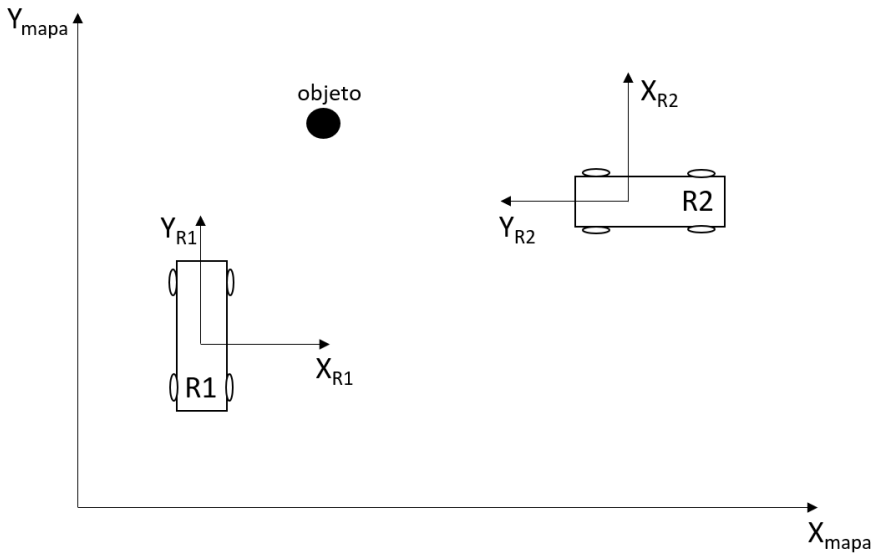


Figura 5.4: Exemplo ilustrativo de referenciais

uma translação, ao referencial em questão, de modo a colocá-lo na orientação e posição desejada. No entanto, um aspeto que merece realce, é o facto de a matriz de rotação possuir características únicas, isto é, esta necessita de respeitar as seguintes propriedades:

- A transposta da matriz de rotação coincide com a sua inversa: $R^T = R^{-1}$;
- Quer as colunas, quer as linhas têm de ser ortogonais entre si;
- Cada coluna é um vetor unitário, assim como cada linha da matriz;
- O determinante da matriz é unitário: $det(R) = 1$;

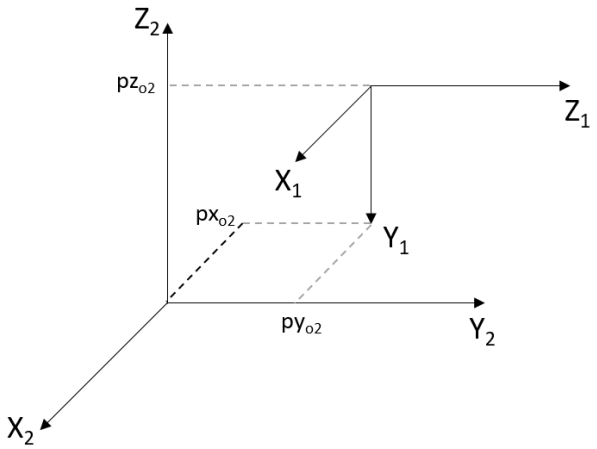


Figura 5.5: Dois Referenciais exemplo

Uma vez definida a matriz de rotação e o vetor de translação que transforma o ponto expresso no referencial 1 no referencial 2 (por exemplo), é possível representar qualquer ponto expresso no referencial 1 (p1) no referencial 2 (p2), de acordo com a equação 5.1.

$$\begin{bmatrix} p2_x \\ p2_y \\ p2_z \end{bmatrix} = R_1^2 \begin{bmatrix} p1_x \\ p1_y \\ p1_z \end{bmatrix} + t_1^2 \tag{5.1}$$

Deste modo, em sistemas de coordenadas 3D, a matriz de rotação que relaciona um referencial com outro é dada pela matriz 5.2, onde:

- $X_1.X_2$ significa a projeção do eixo X do referencial 1, no eixo dos X do referencial 2;
- $Y_1.X_2$ significa a projeção do eixo Y do referencial 1, no eixo dos X do referencial 2;
- $Z_1.X_2$ significa a projeção do eixo Z do referencial 1, no eixo dos X do referencial 2;

  ⋮

- $Z_1.Z_2$ significa a projeção do eixo Z do referencial 1, no eixo dos Z do referencial 2;

Já o vetor de translação (equação 5.3) é obtido projetando a origem do referencial onde o ponto se encontra expresso, no referencial no qual se quer transformar.

$$R_1^2 = \begin{bmatrix} X_1.X_2 & Y_1.X_2 & Z_1.X_2 \\ X_1.Y_2 & Y_1.Y_2 & Z_1.Y_2 \\ X_1.Z_2 & Y_1.Z_2 & Z_1.Z_2 \end{bmatrix} \tag{5.2}$$

$$t_1^2 = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \tag{5.3}$$

A título de exemplo, é apresentado nas equações 5.4 e 5.5, respetivamente, a matriz de rotação e vetor de translação que transforma o referencial 1 no referencial 2 apresentado na figura 5.5.

$$R_1^2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix} \tag{5.4}$$

$$t_1^2 = \begin{bmatrix} px_{o2} \\ py_{o2} \\ pz_{o2} \end{bmatrix} \tag{5.5}$$

5.1.1.1 Referenciais utilizados no robô - ROS

No sistema de localização utilizado na presente dissertação, mais precisamente no sistema de localização do robô Clever, podemos encontrar quatro tipos de referenciais (figura 5.6), tendo cada referencial um propósito específico:

- **Referencial do mundo (*map*)**: referencial fixo utilizado para expressar a posição do robô e dos objetos presentes, em termos de posição absoluta no mundo. Algoritmos de localização global expressam a pose do robô neste mesmo referencial.

- **Referencial do robô (*base_link*)**: referencial móvel que se desloca tendo um ponto escolhido na estrutura do robô, isto é, a origem do referencial é sobreposto a este mesmo ponto. No caso específico dos robôs diferenciais, este ponto é o ponto médio entre as rodas.

- **Referencial do laser *(laser_link)***: referencial móvel que coincide com a posição do laser no robô. As medidas obtidas pelo laser são expressas neste referencial.

- **Referencial da odometria *(odom)***: este é o referencial fixo utilizado para a localização relativa. Por outras palavras, este é um referencial fixo tendo em conta o ponto em que o robô entrou em funcionamento, ou seja, é um referencial no qual é possível obter o deslocamento relativo face ao ponto onde o robô entrou em serviço. Todos os cálculos realizados pela odometria são expressos neste referencial.

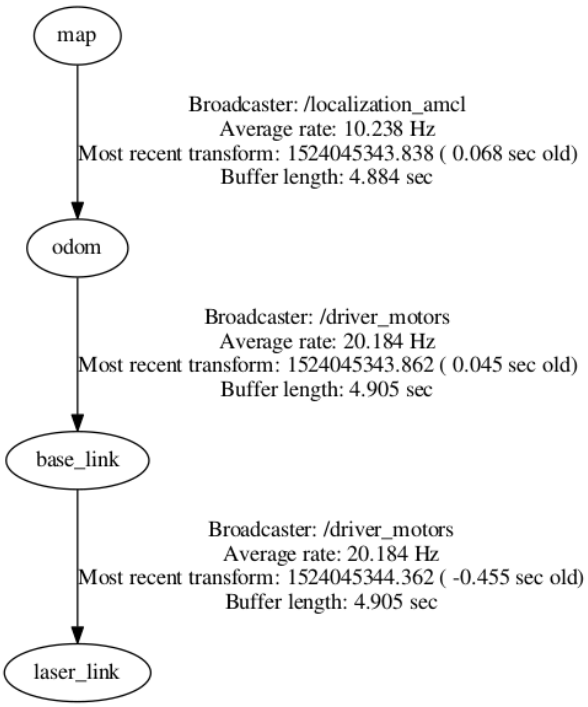
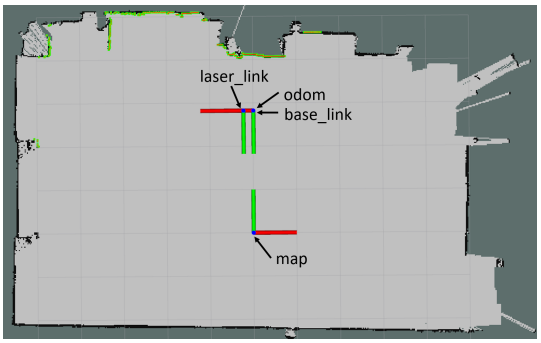
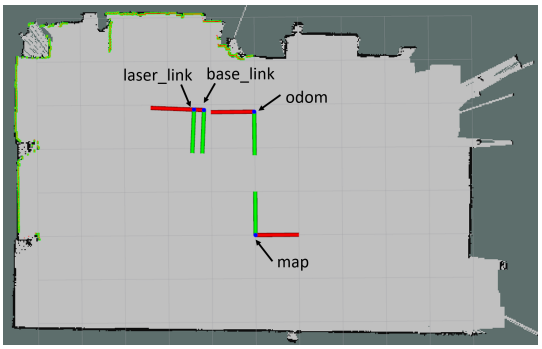


Figura 5.6: Referenciais utilizados no robô Clever

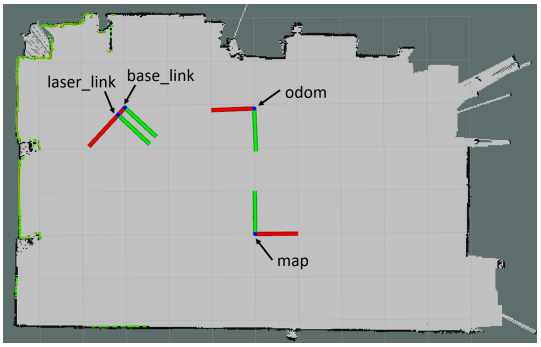
Tal como referido anteriormente, existem dois tipos de referenciais: os fixos (que não se deslocam com o movimento do robô) e os móveis (que se deslocam de acordo com o movimento do robô). Na figura 5.7, encontra-se ilustrado o comportamento dos referenciais presentes no sistema de localização do robô Clever consoante o movimento do mesmo. Tal como se pode constatar pela figura, quando o robô entra em funcionamento o referencial **odom** e **base_link** são coincidentes pois, o referencial **odom** é inicializado tendo em conta a localização inicial do robô. Uma vez que o robô começa a movimentar-se , os referenciais **map** e **odom** não alteram a sua posição ao longo de todo o trajeto do robô. Por outro lado, os referenciais **base_link** e **laser_link** são referenciais móveis, estando estes associados diretamente ao robô, ou seja, estes representam um ponto específico na estrutura do robô, deslocando-se desta forma, em conformidade com o movimento do mesmo.



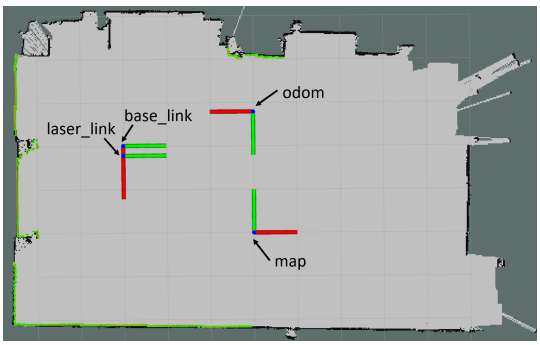
(a) Posição inicial do robô, ou seja, momento no qual o robô entra em funcionamento.



(b) Fase inicial do trajeto do robô (deslocamento linear).



(c) Momento em que o robô efetua uma rotação

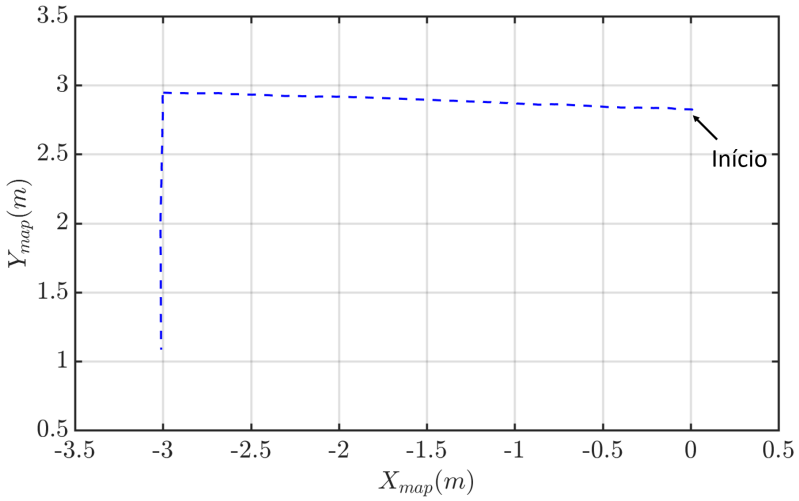


(d) Fase final do trajeto do robô.

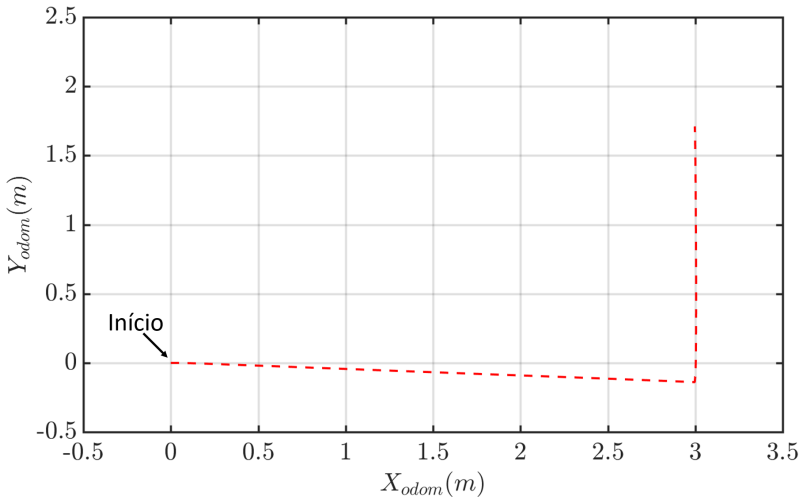
Figura 5.7: Representação do comportamento dos referencias utilizados pelo sistema de localização do robô Clever, tendo em conta o movimento do robô. A cor vermelha no referencial está associada ao eixo X, enquanto que a verde corresponde ao eixo Y.

Para expressar as estimativas de localização do robô, são comummente utilizados os referenciais fixos, pois a sua posição é invariável ao movimento do robô, tal como demonstrado anteriormente. No entanto, a mesma estimativa de localização pode ter coordenadas completamente diferentes, caso seja expressa em referenciais diferentes. Na figura 5.8 está representada essa mesma situação, ou seja, está representada a estimativa de localização do robô (ao longo do seu

percurso), quer no referencial **map** (5.8.a)), quer no referencial **odom** (5.8.b)). Realizando uma análise às figuras, é possível constatar que as coordenadas são completamente diferentes, porém se for analisado o deslocamento relativo face ao ponto inicial, verifica-se que as estimativas são coerentes (ou seja, é a mesma estimativa), embora expressa em referenciais distintos. Portanto, quando são analisados dados é necessário ter em conta o referencial onde estes estão expressos. Caso seja necessário converter os dados para um outro referencial, é necessário recorrer à matriz de rotação e ao vetor de translação que relaciona os referenciais em questão.



(a) Estimativa de localização do robô expressa do referencial **map**.



(b) Estimativa de localização do robô expressa do referencial **odom**.

Figura 5.8: Representação da estimativa de localização do robô em diferentes referenciais.

Embora existam diferentes referenciais, a *framework ROS* fornece, em cada instante, as transformações homogéneas entre cada um dos referenciais, permitindo assim a interação dos vários referencias utilizados em sistemas robóticos.

5.1.2 Correção da localização relativa por parte da localização global

Tal como foi referido na secção 2.1.1, ao deslocamento relativo, nomeadamente à odometria, estão subjacentes diversas fontes de erro, tais como: fontes de erro sistemáticas e não sistemáticas. O facto destes erros estarem presentes na odometria, faz com que a localização dada por este método seja imprecisa, especialmente para grandes distâncias (uma vez que, o erro de localização é proporcional à distância percorrida), inviabilizando a sua utilização como método principal de localização, nos casos em que seja necessário uma localização precisa dos robôs. Porém, caso o robô em questão seja dotado de um outro método de localização mais preciso (*e.g*,localização através de um laser), torna possível a minimização do erro associado à odometria. Mais especificamente, como para distâncias curtas a odometria é bastante precisa, ao existir outro método de localização, permite que seja apenas utilizado os deslocamentos relativos entre cada ciclo de controlo, permitindo eliminar a influência da acumulação de erros inerente à odometria. Desta forma, a informação fornecida pelo deslocamento relativo torna-se mais precisa, viabilizando o seu uso para a fusão com os dados da localização global. A figura 5.9 ilustra o caso onde apenas é usada a odometria como método de localização do robô. Como se pode observar, com o recurso apenas à odometria, não é possível realizar um seguimento da trajetória. No entanto, na figura 5.10 encontra-se representada a situação onde é utilizada não só a odometria, como também, os dados do laser presente no robô. Neste segundo caso como existe um método de localização auxiliar, o robô tem a possibilidade de detetar a sua distância às paredes (quando estas estão ao alcance do laser) permitindo assim corrigir precisamente a sua pose. Uma vez detetada a sua pose, este recorre novamente ao deslocamento relativo para continuar o seu movimento, seguindo assim a trajetória pretendida de uma forma mais precisa.

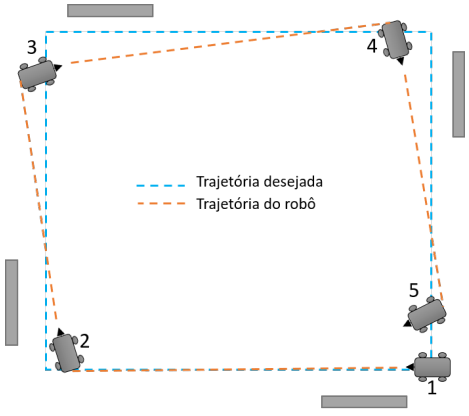


Figura 5.9: Propagação do erro da odometria com a distância percorrida

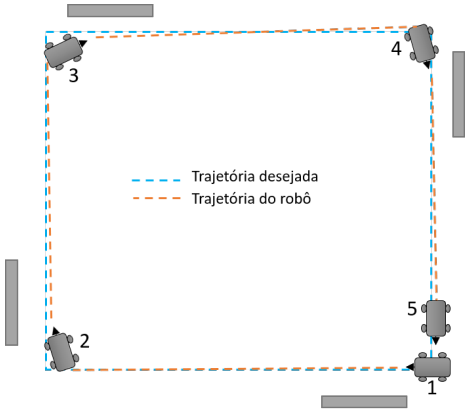


Figura 5.10: Propagação do erro com correção da odometria

5.2 Análise do algoritmo Perfect Match

Tal como demonstrado na revisão bibliográfica (2.1.3), o algoritmo de localização *Perfect Match* (PM) é um algoritmo caracterizado, principalmente, pelo seu baixo peso computacional e

precisão de localização.

No que diz respeito ao algoritmo Perfect Match utilizado na presente dissertação, foi utilizada a versão mais recente do mesmo, tendo sido esta desenvolvida por Sobreira *et al.* [32]. Nesta versão do algoritmo, encontram-se implementadas diversas melhorias face ao proposto inicialmente por Lauer *et al.* [27]. Nomeadamente, o mesmo encontra-se adaptado para uma utilização em ambientes *indoors*; os dados provenientes da odometria são fundidos com os dados sensoriais (com recurso a um filtro de Kalman) para a nova estimação da pose do robô; o critério de paragem do algoritmo de otimização (RPROP) encontra-se alterado (mais especificamente, este deixou de ter um número fixo de interações para passar a ser delimitado por um *threshold* correspondente à convergência do algoritmo e um número máximo de interações, caso o resultado do algoritmo de otimização não seja inferior ao *threshold* definido); e ainda conta com uma fase inicial de pré-processamento dos dados obtidos pelos lasers com o intuito de rejeitar outliers e aumentar a assim a robustez do algoritmo.

Algorithm 1: Pseudocódigo do algoritmo Perfect Match [64]

Input: $\hat{X}(k+1|k), Z_C(k)$
Result: $Z_{Match}(k), P_{Match}(k)$

1 **begin**
2 $X_{init}(k) = \left[\frac{1}{M_{res}} \quad \frac{1}{M_{res}} \quad 1\right] \times \hat{X}(k+1|k)$;
3 **for** *all measures* $Z_{C,i}(k)$ *in* $Z_C(k)$ **do**
4 $Z_{Pix,i}(k) = \left[\frac{1}{M_{res}} \quad \frac{1}{M_{res}}\right] \times Z_{C,i}(k)$;
5 **end**
6 MaxError1 > MaxError2;
7 $Z_{C,F1}(k) = MaxErrorFilter(Z_{Pix}(k), X_{init}(k), MaxError1)$;
8 $X_{F1} = OptimizeMatch(Z_{C,F1}(k), X_{init}(k))$;
9 $Z_{C,F2}(k) = MaxErrorFilter(Z_{C,F1}(k), X_{F1}(k), MaxError2)$;
10 $X_{F2} = OptimizeMatch(Z_{C,F2}(k), X_{F1}(k))$;
11 $P_{Match}(k) = CovarianceCalculations(Z_{C,F2}(k), X_{F2}(k))$;
12 $Z_{Match}(k) = \left[\frac{1}{M_{res}} \quad \frac{1}{M_{res}} \quad 1\right] \times X_{F2}(k)$;
13 **end**

No Algoritmo 1, encontra-se, sob a forma de pseudocódigo, a essência do algoritmo de localização utilizado, onde, como entradas possui a posição do robô (tendo já em conta o processamento de dados da odometria - $\hat{X}(k+1|k)$), e ainda os dados sensoriais obtidos ($Z_C(k)$). Como saídas retorna a estimativa da posição do robô no mundo ($Z_{Match}(k)$), bem como a matriz de covariância ($P_{Match}(k)$) que quantifica a incerteza da estimativa da posição do robô. No algoritmo em questão, é aplicada primeiramente uma pré-filtragem (Linha 7), calculando a estimativa de localização posteriormente (Linha 8). De seguida, é aplicada novamente uma filtragem, porém mais exigente (Linha 9), sendo calculada uma nova estimativa para os dados completamente filtrados (Linha 10). Ao ser aplicado este tipo de filtragem permite, numa primeira instância, eliminar dados do laser

que sejam claramente *outliers* (tendo em conta um erro de posição do robô, daí o *threshold* ser menos exigente), para numa segunda instância, contendo uma melhor estimativa do robô, aplicar um segundo filtro (desta vez mais exigente) eliminando os restantes outliers presentes nos dados do laser. Desta forma, o algoritmo torna-se claramente mais robusto na presença de *outliers* [64]. Por fim é calculada a matriz de covariância que quantifica a incerteza da estimação (linha 11). As linhas 2, 4 e 12 estão associadas a uma otimização do peso computacional. Como o algoritmo é um algoritmo que utiliza um mapa de ocupação em grelha, a pose do robô e as medidas provenientes dos sensores são escaladas de acordo com a resolução do mesmo (M_{res}) (linhas 2 e 4, respetivamente), permitindo assim que uma posição no referencial global seja convertida na célula do mapa respetiva. No final do algoritmo, o resultado da otimização é novamente convertido para o referencial global (linha 12).

Como é percetível pela análise do Algoritmo 1, o cerne do algoritmo de localização Perfect Match é o seu algoritmo de otimização - RPROP. Este algoritmo, que foi desenvolvido para problemas não lineares, tem por base a minimização do erro de *matching* baseado nas primeiras derivadas da função custo utilizada (Figura 2.8). Por outras palavras, o objetivo do algoritmo RPROP é, consoante os dados sensoriais obtidos, verificar qual a posição no mapa que o robô detém, que minimiza o erro de *matching* entre os dados. Desta forma, a posição que minimiza o erro, teoricamente, é a posição real do robô no meio ambiente.

5.2.1 Testes realizados e respetiva análise

Associado à execução do PM, existem diversos parâmetros configuráveis. Através do ajuste destas mesmas configurações (tendo como ponto de partida os resultados obtidos no estudo precedente a esta dissertação [63]), as que melhor se ajustam ao robô Clever para a execução do algoritmo encontra-se representada na Tabela 5.1, tendo sido estas as configurações utilizadas para os testes aplicados para a análise deste mesmo algoritmo. Um aspeto que merece realce, é o facto de se ter desativado os critérios de paragem do algoritmo RPROP, com o objetivo de forçar o PM a devolver a melhor estimativa possível em cada iteração, isto é, ou a convergência total do algoritmo, ou a utilização do número máximo de iterações ($NumIterationMax$).

5.2.1.1 Ambiente de navegação com *outliers*

Tal como referido na introdução do presente capítulo, os testes realizados com outliers serviram para testar diversos cenários correspondentes à navegação do robô no mundo real, com o objetivo de perceber em que situações, o algoritmo *Perfect Match*, apresentava um mau funcionamento.

Na navegação do robô é comum estarem presentes objetos não mapeados (*outliers*) no seu trajeto. Um ambiente industrial é caracterizado pelas suas constantes alterações, isto é, apenas se mantêm constantes as estruturas das próprias instalações sendo o restante sujeito a movimentações. Já em ambientes não industriais (como por exemplo, um corredor de um hospital ou de uma faculdade), é comum os objetos se manterem constantes, no entanto, existe um fluxo de pessoas

Tabela 5.1: Configurações do Perfect Match utilizadas

LaserMeasuresNum	726
NumIterationMax	150
OptimizationStopTranslation	0.0
OptimizationStopRotation	0.0
MinPercentValideSensorData	0.0 %
DeltaUpdateInitPosition	0.01 m
DeltaUpdateInitOrientation	0.001 graus
Lc	0.1
DriftRotationDeltaRot	0.001
DriftTranslationDeltaD	0.001
DriftTranslationDeltaRot	0.0003
SensorsKErrorTheta	100.0
SensorsKErrorXY	10.0
StateMinVarTheta	0.001
StateMinVarXY	0.01
OperationMode	SensorsOdomFusion
MaxErrordistFilterEnabled1	true
”LaserFilterMaxDist1”	0.5 m
”MaxErrordistFilterEnabled2”	true
”LaserFilterMaxDist2”	0.05 m
”WeightenMeasureDistEnable”	true

muito grande, tendo um efeito semelhante à movimentação de objetos num ambiente industrial. Como a presença de outliers limita o campo de visão do robô, estes podem obstruir uma zona do mapa que é fulcral para a sua boa localização. Na figura 5.11, encontra-se representada essa mesma situação. O cenário representado na figura pretende simular a presença de um objeto ou um conjunto de pessoas a obstruir a deteção de um zona mapeada (neste caso uma parede) no percurso do robô.

(a) (b)

Figura 5.11: Obstrução de uma estrutura do mapa presente no robô - cenário real de testes

Assim colocou-se o robô a executar a trajetória predefinida no ambiente referido anteriormente. Tal como é visível na figura 5.12.a) e 5.12.b), a estimativa de localização do robô dada

pelo algoritmo PM durante o percurso inicial, não apresenta erro de localização (apenas possui um pequeno desvio face à localização de referencia justificado pelo controlador utilizado). Isto deve-se ao facto de o robô possuir uma boa informação sobre o meio envolvente, permitindo assim rejeitar o outlier quando este é detetado. No entanto, como é possível constatar através da análise do mapa presente no robô, na última parte do percurso efetuado pelo mesmo (figura 5.12.c)), a parede mapeada será a única referência para o algoritmo de otimização associado ao *Perfect Match*. Como esta se encontra obstruída por um objeto, o algoritmo de otimização - RPROP - começa a convergir para a zona do mapa que melhor se ajusta aos pontos detetados. Desta forma o robô deixa de considerar o objeto externo ao mapa como outlier considerando-o como uma parede mapeada, convergindo assim para uma posição errada, que no presente caso é uma localização mais próxima da parede mapeada (figura 5.12.c).7 e 5.12.c).8). Após ultrapassar o obstáculo, o algoritmo volta a convergir para a pose correta, tal como representado na figura 5.12.d). Na figura 5.13, é possível comparar a estimação de localização dada pelo PM ao longo de todo o trajeto do robô, face à localização de referência utilizada.

Analisando os resultados obtidos, é possível verificar que quando é obstruída uma zona do mapa e o robô se encontra perto da mesma, o algoritmo *Perfect Match* falha, convergindo para uma pose errada. No entanto, ao analisar-se os dados provenientes da odometria do robô (figura 5.14), verifica-se que, embora os dados estejam expressos num referencial diferente, não existe nenhum "salto" na localização do robô, contrariando o resultado da localização dada pelo algoritmo PM (que é a fusão entre os dados provenientes da odometria com os dados do laser).

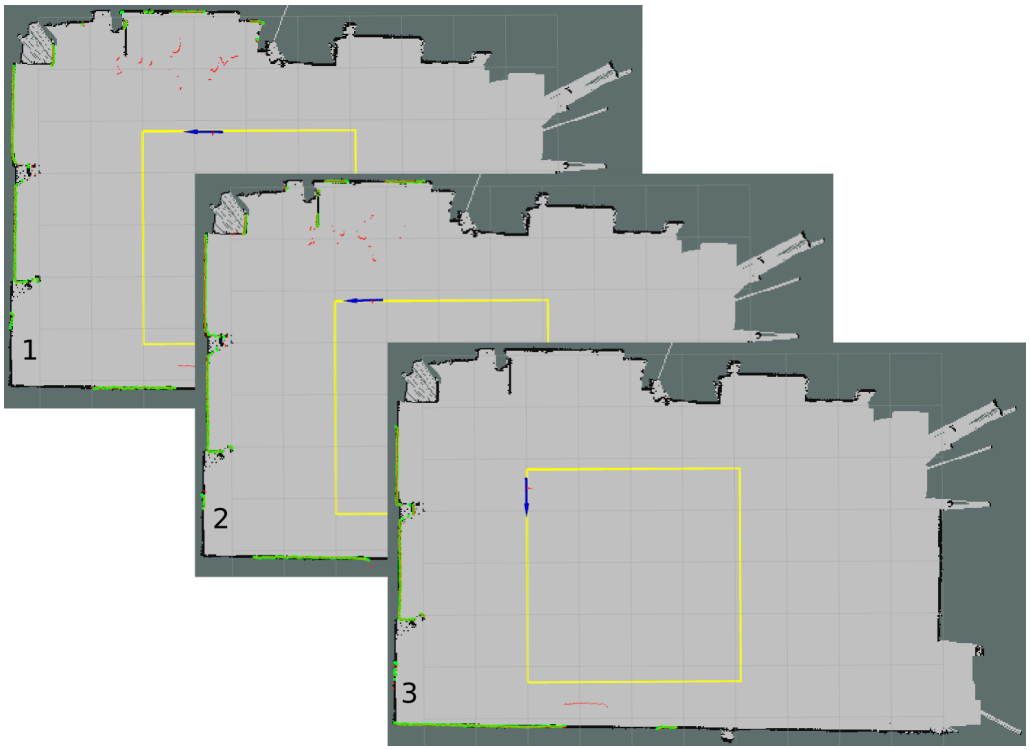


Figura 5.12: a) - Localização dada pelo PM (a azul) na primeira secção do percurso face à localização pretendida (a amarelo).

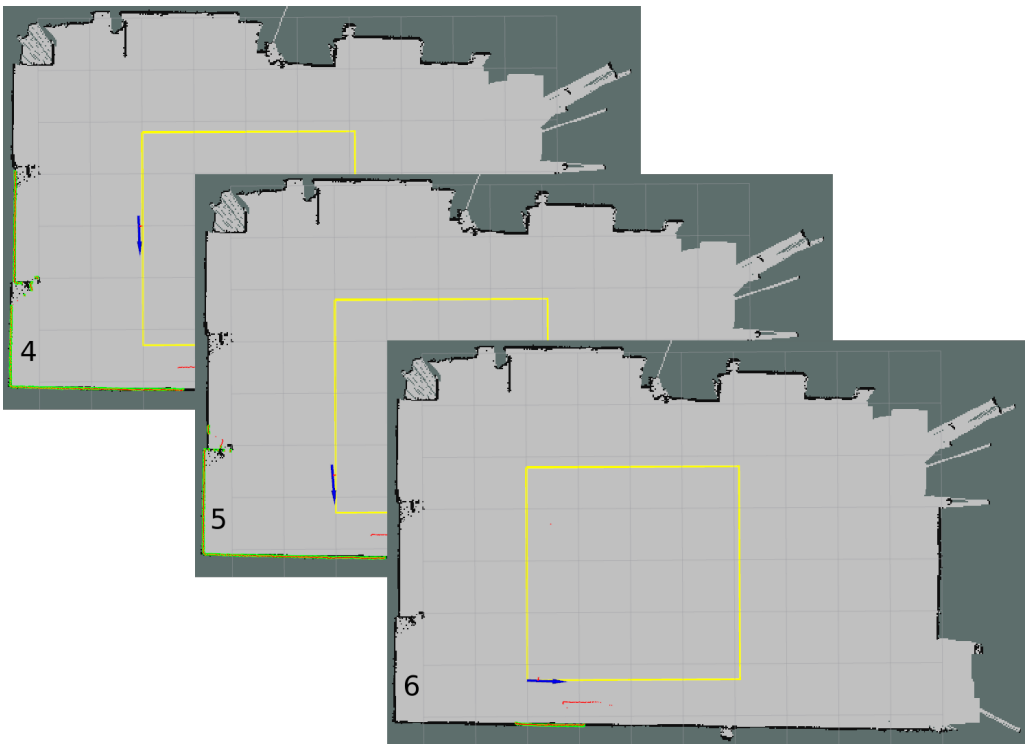


Figura 5.12: b) - Localização dada pelo PM (a azul) na segunda secção do percurso face à localização pretendida (a amarelo).

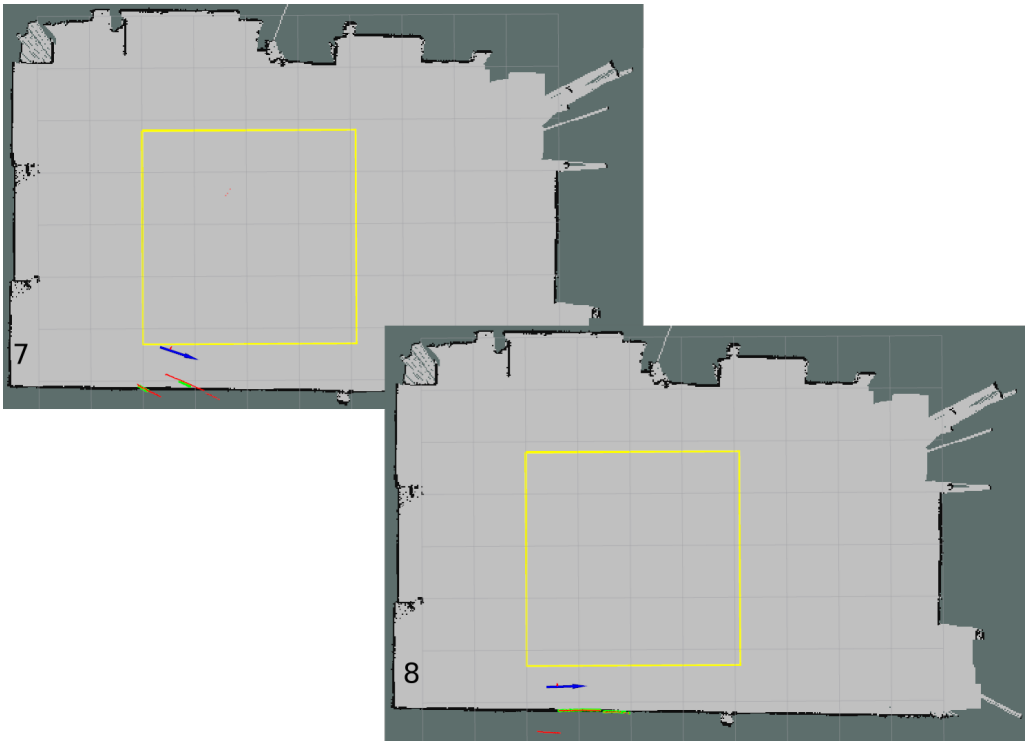


Figura 5.12: c) - Localização dada pelo PM (a azul) na última secção do percurso do robô, reagindo à presença de *outliers*.

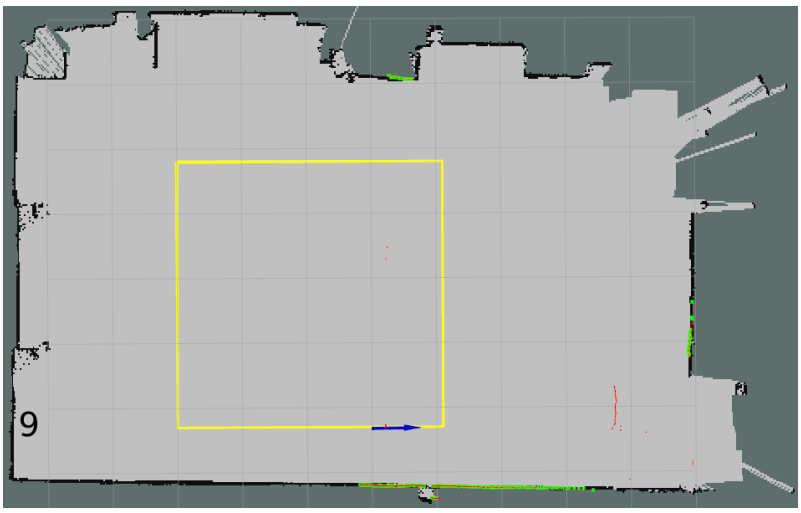


Figura 5.12: d) - Localização dada pelo PM (a azul) na última secção do percurso do robô: momento de recuperação da localização por parte do PM.

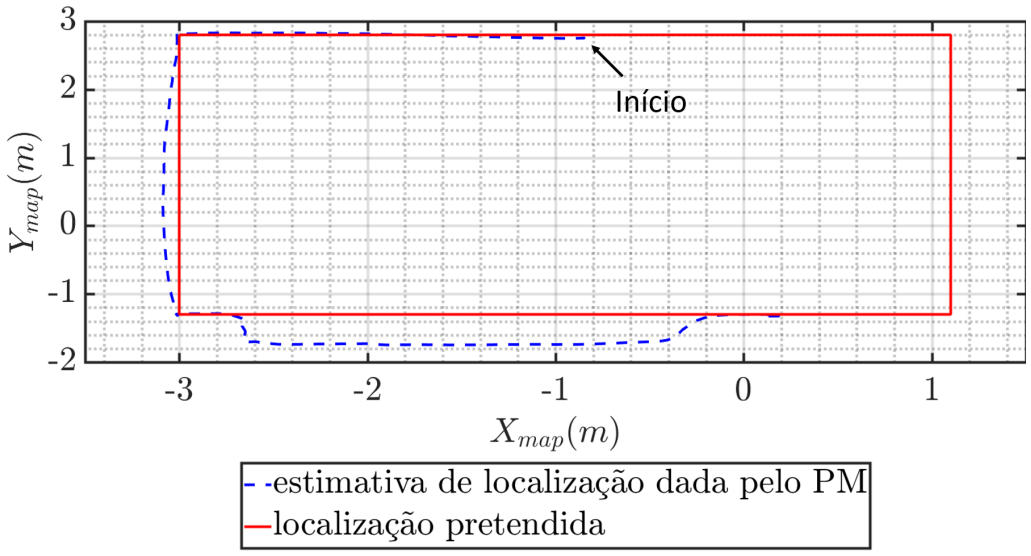


Figura 5.13: Gráfico da estimativa localização dada pelo algoritmo PM, face à localização pretendida, expressa no referencial **map**. Na parte final do percurso é visível uma falha na estimativa de localização do PM.

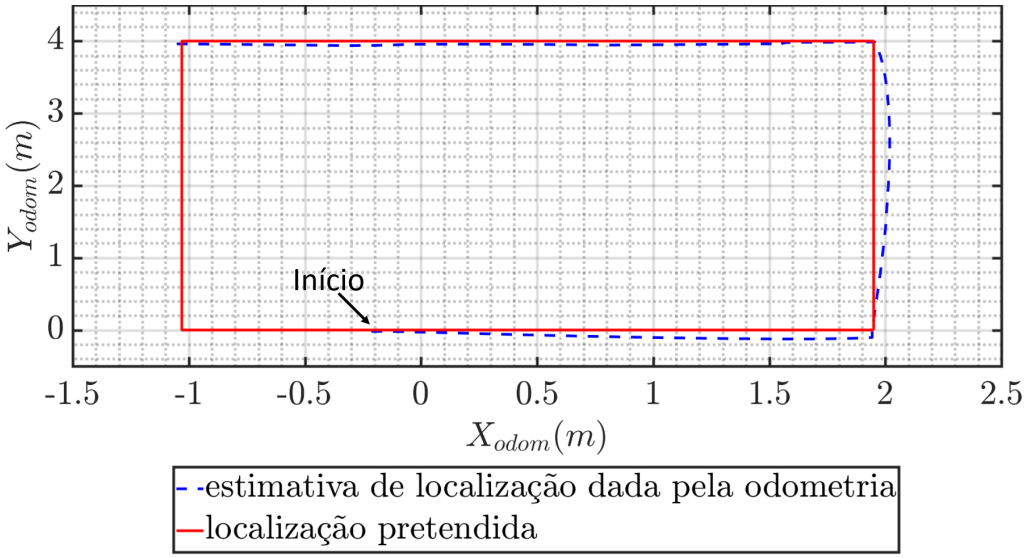


Figura 5.14: Odometria do robô no trajeto efetuado face à localização pretendida, expressa no referencial **odom**. Tal como é visível na figura, a odometria não corrobora o "salto" apresentado pela estimativa de localização do PM.

5.2.1.2 Erros forçados na odometria

Relativamente aos erros introduzidos, estes consistiram na elevação de uma das rodas do robô, simulando a patinagem das mesmas, o qual provocará uma indicação errada no deslocamento relativo do robô.

No que diz respeito a erros provenientes da estimativa do deslocamento relativo, que no presente caso trata-se exclusivamente da odometria, o algoritmo *Perfect Macth* provou ser bastante robusto, mesmo com grandes erros neste tipo de estimativa. Tal como é visível na figura 5.15, o algoritmo não divergiu da verdadeira pose do robô, mesmo com fortes erros introduzidos na odometria tal como demonstra a figura 5.16.

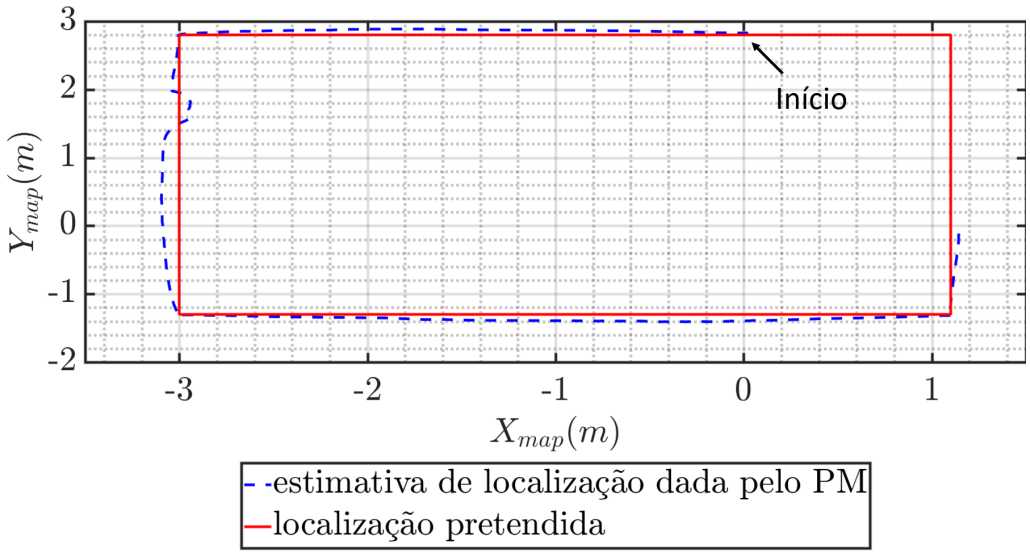


Figura 5.15: Gráfico da estimativa localização dada pelo algoritmo PM, face à trajetória de referência utilizada, expressa no referencial **map**. Tal como é visível na figura, a estimativa dada pelo algoritmo não diverge da trajetória pretendida.

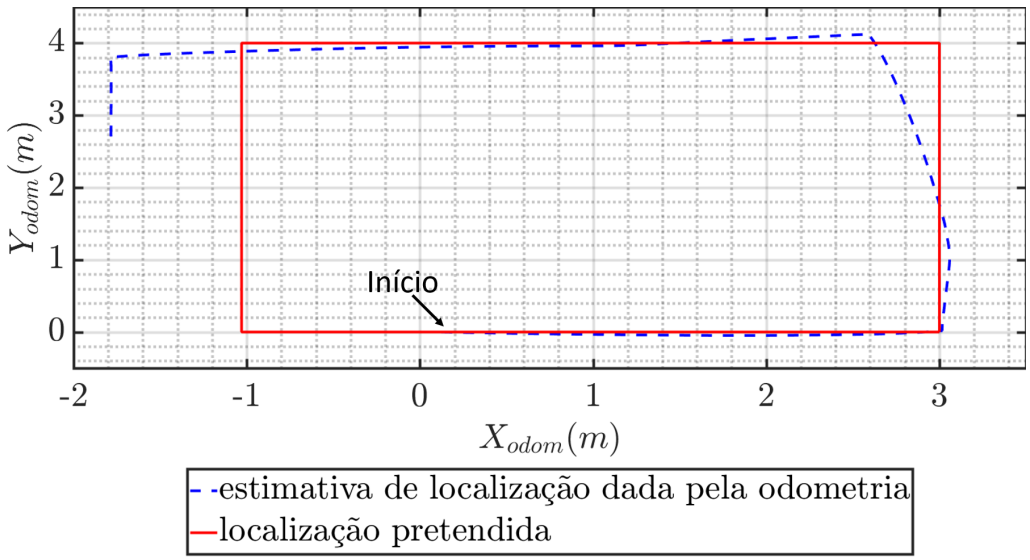


Figura 5.16: Estimativa dada pela odometria do robô no trajeto efetuado face à localização de referência utilizada, expressa no referencial **odom**. Quando são forçados erros na odometria (nomeadamente, a elevação do robô), a estimativa dada pela odometria diverge da localização pretendida.

5.3 Análise do algoritmo de localização AMCL

Das várias vertentes do algoritmo MCL referidas na secção 2.1.2.2, a utilizada na presente dissertação, é a versão disponibilizada pelo ROS, a qual implementa diversas configurações do algoritmo de localização sendo esta denominada por AMCL - *Adaptative Monte Carlo Localization*. Esta versão do algoritmo, também conhecida como KLD-MCL (proposto por Fox [20]), implementa, para além da funcionalidade que permite um ajuste dinâmico do número partículas conforme o grau de certeza da localização do robô, as melhorias propostas por Thrun *et al* [8] (cuja versão do algoritmo é conhecida por *Augmented MCL*). Por outras palavras, o algoritmo utilizado tem a capacidade de adaptar o número de partículas dinamicamente (apenas são lançadas as necessárias para que seja possível localizar o robô), evitando assim o uso de partículas "desnecessárias" e, portanto, diminuindo o peso computacional. Contudo, são injetadas novas partículas aleatórias a cada iteração para existir a possibilidade de recuperar de situações de "rapto" do robô. Com o intuito de modelizar o comportamento dos lasers, esta versão do algoritmo, disponibiliza dois algoritmos para modelizar o comportamento destes. Nomeadamente, são implementados os algoritmos *"Beam Range Finder"* (onde é calculada a probabilidade de obter determinada medida do laser em cada varrimento - médodo conhecido como *"ray casting"*) e "Likelihood Field" (o qual projeta os dados no referencial mapa, com o intuito de verificar a sua viabilidade espacial e assim validar as medidas do laser). De acordo com o funcionamento do algoritmo e com a finalidade de evitar a perda de localização do robô, este apenas é executado aquando a movimentação do robô, ou seja, quando o mesmo está imóvel o algoritmo não é executado para que as partículas não se concentrem exclusivamente num local, garantindo a robustez do algoritmo. Por conseguinte, existem diversos parâmetros configuráveis associados a cada uma das funcionalidades do algoritmo, estando estes representados na tabela 5.2, bem como os valores utilizados na sua aplicação no robô Clever (os quais foram estabelecidos no estudo precedente a esta dissertação).

Analisando o algoritmo presente na figura 2.2 (o qual é o cerne das todas as versões desenvolvidas do algoritmo MCL), é possível, e tal como referido na secção 2.1.2.2, decompor o algoritmo em 3 fases distintas. A primeira fase (fase de predição), corresponde à fase onde são propagadas as partículas segundo o modelo de movimentação do robô (nomeadamente a odometria), é uma fase com elevada importância para o funcionamento do algoritmo. Posteriormente na segunda fase, são utilizados os dados provenientes dos lasers para a validação e atribuição dos pesos às partículas. A última fase corresponde apenas à reamostragem das partículas no mapa de acordo com o resultado das fases anteriores. Portanto, realizando uma análise preliminar ao algoritmo, é possível que este dê mais ênfase aos dados oriundos da odometria face aos dados provenientes dos sensores, uma vez que é com os dados da odometria que existe uma propagação das partículas não utilizando os dados do laser para tal (primeira fase).

5.3.1 Testes realizados e respetiva análise

Para os testes realizados ao algoritmo AMCL, foram utilizadas as configurações apresentadas na tabela 5.2. Em termos de cenários de testes, foram utilizados os mesmos cenários que os

Tabela 5.2: Configurações do algoritmo AMCL utilizadas

update_min_d	0.1 m
update_min_a	0.17453 rad
resample_interval	1.0
kld_err	0.01
kld_z	0.99
recovery_alpha_slow	0.0
recovery_alpha_fast	0.0
initial_cov_xx	0.0
initial_cov_yy	0.0
initial_cov_aa	0.0
laser_model_type	“likelihood_field”
laser_min_range	-1.0
laser_max_range	-1.0
laser_max_beams	100.0
laser_z_short	0.0
laser_z_max	0.0
laser_sigma_hit	0.1
laser_likelihood_max_dist	2.0 m
laser_lambda_short	0.1
min_particles	400
max_particles	400
laser_z_hit	0.85
laser_z_rand	0.15
odom_alpha1	0.2
odom_alpha2	0.2
odom_alpha3	0.2
odom_alpha4	0.2

utilizados no algoritmo PM. Nomeadamente, para o teste do algoritmo na presença de *outliers*, foi utilizado o cenário ilustrado na figura 5.11. Para os testes realizados com o intuito de analisar a robustez do algoritmo face aos dados da odometria, foram introduzidos erros semelhantes aos introduzidos aquando do teste do algoritmo PM.

5.3.1.1 Ambiente de navegação com *outliers*

No que diz respeito à navegação na presença de *outliers*, o algoritmo AMCL provou ser mais robusto do que o algoritmo *Perfect Match*. De acordo com os testes aplicados ao algoritmo e tal como se pode observar na figura 5.17, a presença destes não implicou uma divergência do algoritmo, dado que o mesmo continuou a fazer uma bom *tracking* da pose do robô. No entanto a presença do *outilier* na última secção do percurso do robô, aumentou o grau de incerteza do estimativa dada pelo algoritmo, tal como é possível constatar na figura 5.18 pelo aumento da dispersão do número de partículas no momento da passagem pelo *outlier*. Uma vez ultrapassado (figura 5.18.3), a dispersão das partículas diminui, concentrando-se na verdadeira pose do robô.

Caso a presença do outlier fosse mais prolongada, as partículas dispersar-se-iam cada vez mais, podendo-se concentrar numa pose errada do robô e provocar assim uma falha no algoritmo.

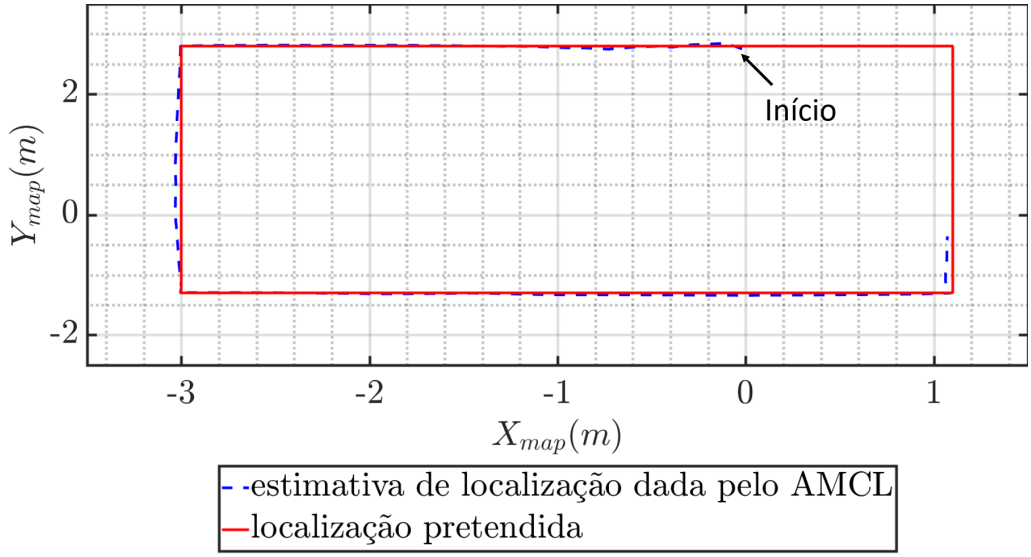


Figura 5.17: Gráfico da estimativa localização dada pelo algoritmo AMCL, face à trajetória de referência utilizada, expressa no referencial **map**. De acordo com a figura, o algoritmo estimou corretamente a localização do robô ao longo de todo o seu percurso.

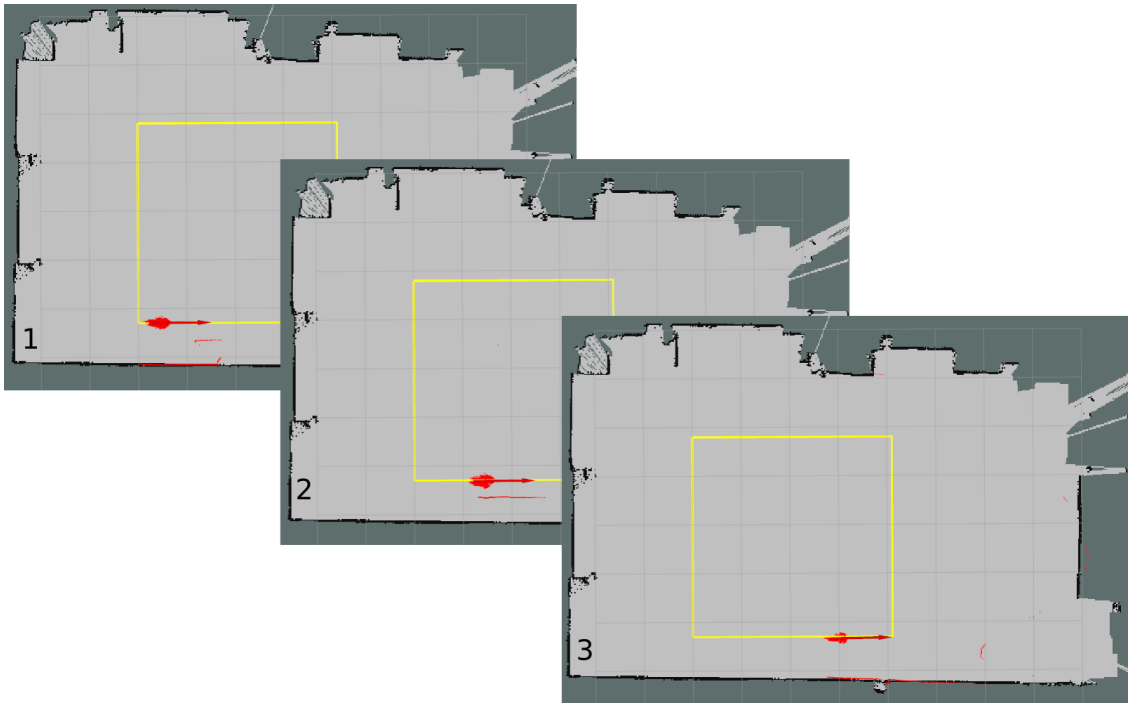


Figura 5.18: Evolução da dispersão das partículas utilizadas pelo algoritmo AMCL aquando a presença do *outlier*

5.3.1.2 Erros forçados na odometria

Apesar do AMCL ser um algoritmo muito robusto no que diz respeito à localização na presença de outliers, o algoritmo revelou ser sensível aos dados provenientes da odometria. Como previsto na análise realizada na secção 5.3, a odometria possui um peso relevante face aos dados do laser, uma vez que utiliza a mesma na primeira etapa do algoritmo, isto é, na propagação da estimativa de localização . No caso dos dados provenientes da odometria estarem errados, existirá uma má propagação da estimativa da pose que, consequentemente, culminará numa perda de localização. A figura 5.19 representa esta mesma situação. Um erro na odometria (nomeadamente, a patinagem das rodas - indicando um maior deslocamento do que o que foi realizado verdadeiramente - figura 5.20), provoca uma má estimativa no algoritmo AMCL, dado que este continuou a propagar a estimativa, quando na realidade o robô encontrava-se parado, resultando finalmente numa perda do *tracking* da pose do robô.

Como no algoritmo AMCL a odometria possui uma grande influência, o método utilizado para identificação da localização aproximada do robô, para este teste, foi modificado. Mais especificamente, e após ter sido comprovada a grande robustez do algoritmo *Perfect Match* face a uma má estimativa do deslocamento relativo do robô, o mesmo foi utilizado como indicador da posição aproximada do robô. Ou seja, a posição aproximada do robô encontra-se representada como a estimativa fornecida pelo *Perfect Match* ao contrário dos testes realizados anteriormente.

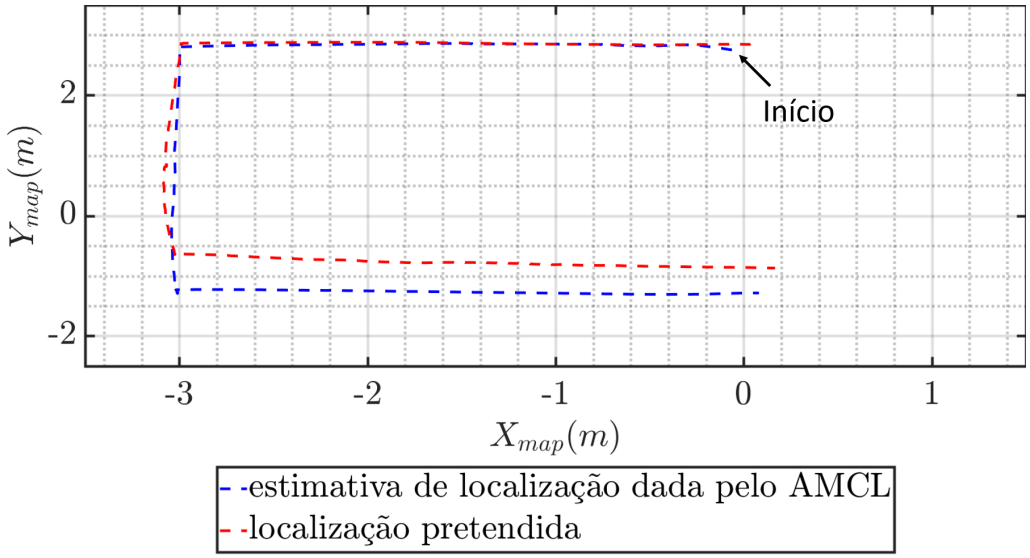


Figura 5.19: Gráfico representante da estimativa de localização dada pelo algoritmo AMCL, face à trajetória pretendida, expressa no referencial **map**. Devido aos erros introduzidos pela odometria, o algoritmo AMCL propagou demasiado a sua estimativa, indicando um maior deslocamento face ao realmente realizado.

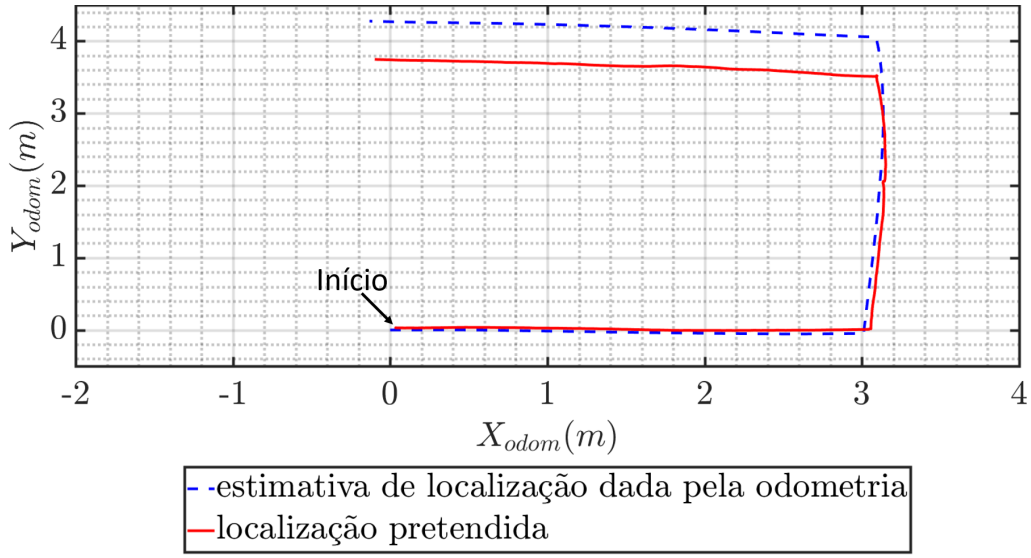


Figura 5.20: Estimativa dada pela odometria do robô ao longo do seu percurso, expressa no referencial **odom**. Com a elevação do robô (e consequente patinagem das rodas do mesmo), é indicado um maior deslocamento relativo face ao realmente realizado.

5.4 Análise ao Supervisor proposto por Farias *et al.* [38]

Uma das técnicas presentes na comunidade científica com o intuito de obter uma localização fiável e robusta, passa pela utilização de múltiplos algoritmos de localização em simultâneo, onde a coordenação destes é realizada por um supervisor. Assim é possível ter, duas ou mais, estimativas da localização do robô dadas por diferentes algoritmos conferindo, ao supervisor, a possibilidade de comparar essas estimativas e tomar decisões de acordo com indicadores de desempenho fornecidos pelos próprios algoritmos de localização.

Segundo este principio, e tal como referido na secção 2.2, Farias *et al.* [38] desenvolveram um supervisor de algoritmos de localização que utiliza a estimativa da pose de dois algoritmos de localização. Mais precisamente, o supervisor (cujo funcionamento está descrito sob a forma de um diagrama de atividade na figura 5.21) utiliza a estimativa fornecida pelo algoritmo AMCL e pelo *Perfect Match*. De acordo com o diagrama de atividade presente na figura 5.21, o algoritmo começa primeiramente por comparar as poses associadas a cada um dos algoritmos. Caso estas estejam coerentes, isto é, caso a pose dada pelos dois algoritmos seja muito semelhante, a escolhida pelo supervisor é a dada pelo PM. Caso contrário, são avaliados, quer o indicador de desempenho dado pelo PM, quer o indicador de desempenho associado ao AMCL. Se o índice associado ao PM for aceitável a pose escolhida pelo algoritmo de supervisão é a dada pelo mesmo, independentemente do resultado da análise do indicador alusivo ao AMCL. Portanto, o supervisor desenvolvido pelos autores, privilegia a pose fornecida pelo *Perfect Match* com a justificação de este ser mais preciso, quando comparado com o algoritmo AMCL [38]. No caso da pose dada

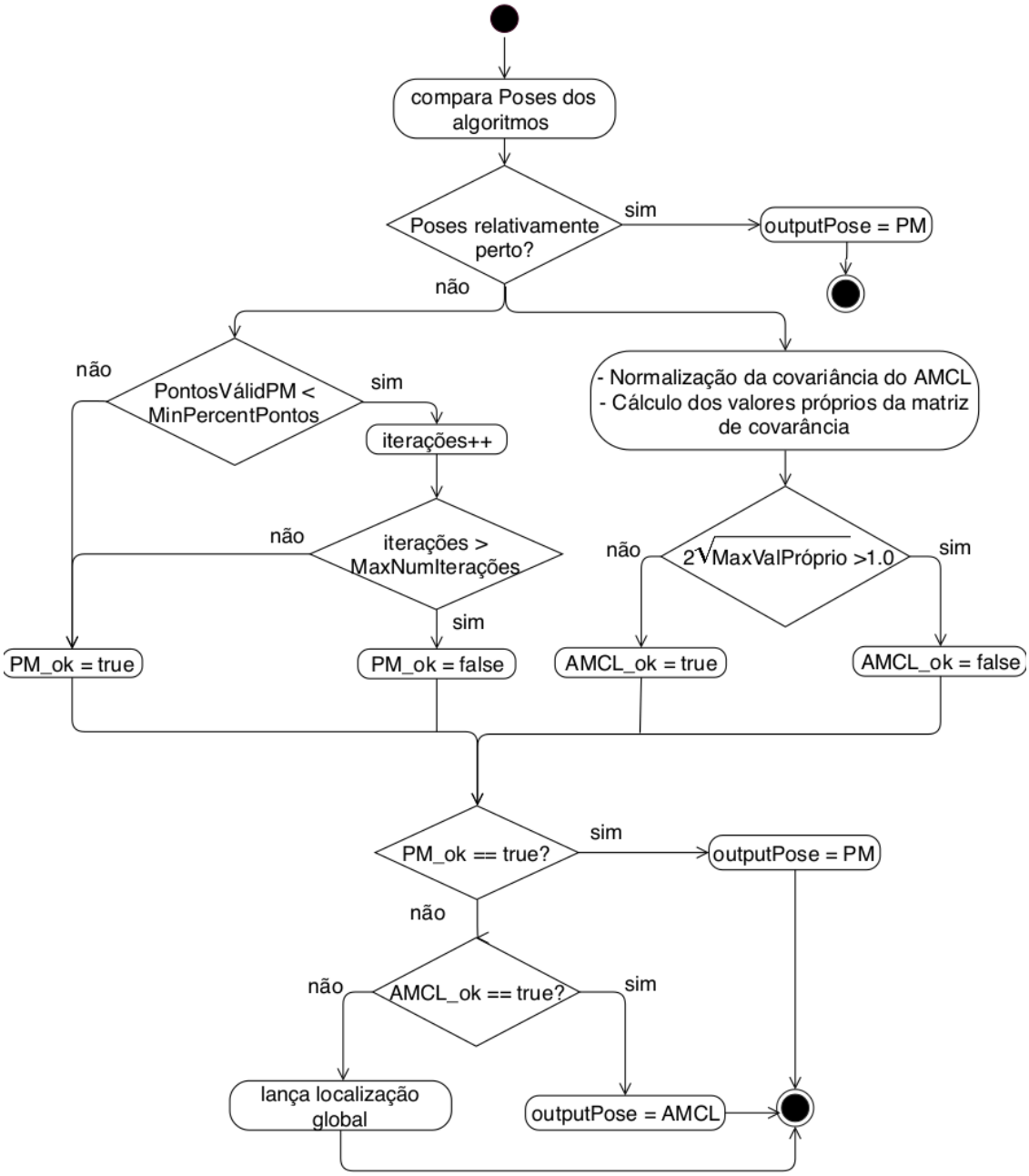


Figura 5.21: Diagrama de atividade do supervisor proposto por Farias *et al.* [38]

pelo *Perfect Match* não ser credível, é então utilizada a pose fornecida pelo AMCL se o índice de desempenho deste assim o permitir. No caso dos dois algoritmos possuírem uma má estimativa de localização, os autores referem que, o supervisor ativa um mecanismo de localização global com o intuito de recuperar a posição do robô, porém não o especificam.

Realizando uma análise do algoritmo referente ao supervisor proposto de Farias *et al.* [38], é possível constatar que a essência do mesmo consiste nos indicadores de desempenho associados a cada um dos algoritmos, uma vez que são estes os fatores decisivos para a validação das poses fornecidas pelos algoritmos. No caso do *Perfect Match* é utilizado o número de pontos considerados

como corretos pelo algoritmo (Figuras 5.22 e 5.23), com a justificação de que quando o robô está bem localizado, o número de pontos do laser aceites pelo algoritmo é elevado (pelo facto de estes estarem de acordo com o mapa presente no robô). Já o algoritmo AMCL recorre ao máximo valor próprio da matriz de covariância associada à estimativa da pose dada pelo mesmo. Os autores ao analisarem o máximo valor próprio da matriz de covariância, estão a analisar o maior raio da elipsoide associada à covariância de estimação. Como a covariância representa a incerteza da estimação da pose, os autores analisam assim a direção com maior incerteza. Caso a raiz quadrada do máximo valor próprio (da matriz de covariância normalizada) seja superior a duas vezes o desvio padrão, é sinónimo de uma grande incerteza na posição do robô, descredibilizando assim a pose dada pelo algoritmo AMCL.

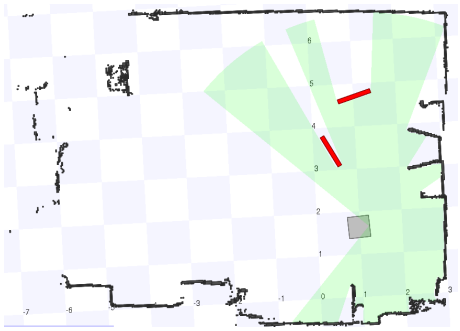


Figura 5.22: Situação de navegação simulada (os objetos vermelhos correspondem a *outliers*) - *Simulator Stage*

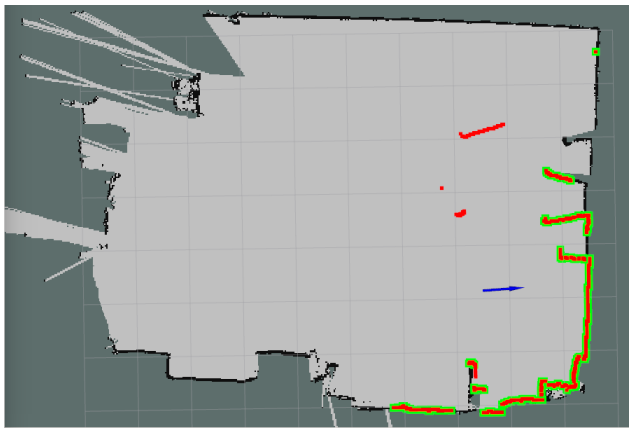


Figura 5.23: Mapa obtido através da ferramenta *RVIZ*. A azul: pose estimada pelo algoritmo PM; a vermelho: todos os pontos detetados pelo laser; a verde: pontos do laser aceites pelo algoritmo de localização

5.4.1 Testes realizados e respetiva análise

Para a execução do supervisor desenvolvido por Farias *et al.* existem diversos parâmetros configuráveis. Para a análise e testes aplicados ao mesmo, foram utilizadas as configurações presentes na tabela 5.3. Assim sendo, o supervisor foi preparado de modo a supervisionar os algoritmos *Perfect Match* e AMCL a uma taxa de 10 Hz fundindo os seus resultados e tomando medidas corretivas, caso necessário. Relativamente aos testes aplicados, estes foram os mesmos que os aplicados aquando da análise individual dos algoritmos de localização.

5.4.1.1 Ambiente de navegação com *outliers*

Com o intuito de verificar se o supervisor desenvolvido por Farias *et al.* [38] era capaz de solucionar o problema detetado aquando da análise do algoritmo *Perfect Match*, nomeadamente na presença de *outliers*, submeteu-se o supervisor ao mesmo cenário de teste (figura 5.11) apresentando este, os resultados representados na figura 5.24. Analisando os resultados obtidos,

Tabela 5.3: Configurações associadas ao Supervisor de Farias *et al.* [38]

LoopFrequency	10 Hz
SupervisionMode	Fusion
DefaultAlgorithm	"perfect_match"
AlgorithmsToUse	"perfect_match amcl"
CorrectPose	true
PoseDistanceErrorThreshold	0.2
PoseOrientationErrorThreshold	15.0 °
MaxAmclDistanceErrorAllowed	0.05 m
MaxAmclOrientationErrorAllowed	20
AmclCovarianceThreshold	0.5
MaxNumberOfCyclesAllowingHighErrors	10.0
MinMatchingErrorThreshold	0.05
MinPercentageLaserPoints	70

constata-se que o supervisor não superou o problema detetado anteriormente, isto é, o supervisor não identificou que o *Perfect Match* perdeu o *tracking* da pose do robô, culminando assim numa falha de localização. De acordo com a análise realizada na secção 5.4, o supervisor desenvolvido pelos autores utiliza o número de pontos do laser considerados como corretos pelo algoritmo *Perfect Match*. Por este facto, quando o algoritmo converge erradamente para uma posição, este converge para a posição que minimiza a função custo do mesmo, que consequentemente é a que maximiza o número de pontos do laser aceites pelo algoritmo. Assim ao convergir para uma pose errada devido à presença de *outliers*, os dados aceites pelo laser aumentam, tal como demonstra a figura 5.25 (tempo relativo = 720). Ao aumentar esta percentagem, o algoritmo passa a ter um bom indicador de desempenho, fornecendo a indicação que a estimativa de localização se encontra correta. Como as poses dadas pelos dois algoritmos de localização diferem, são analisados os indicadores de desempenho e uma vez que o indicador associado ao algoritmo *Perfect Match* indica que este possui uma boa estimativa de localização, o supervisor toma a medida corretiva de corrigir a pose dada pelo AMCL em função da pose fornecida pelo PM. Desta forma, embora o AMCL continuasse bem localizado, o supervisor forçou o mesmo para a posição dada pelo PM, tendo esta medida comprometido a localização do robô, uma vez que esta ação levou à perda do *tracking* da pose.

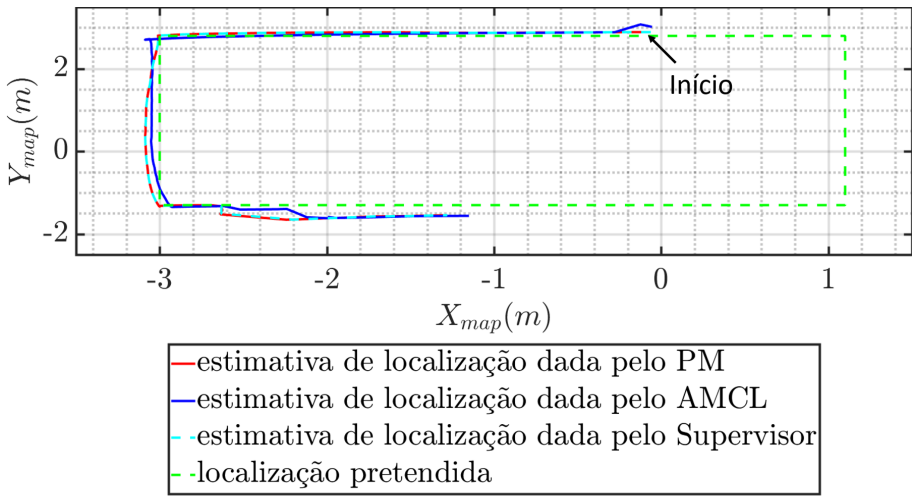


Figura 5.24: Problema de localização detetado no supervisor de Farias *et al.* [38]. Mais precisamente, indicação de boa localização do PM, quando na realidade este convergiu para uma pose errada, corrigindo erradamente a estimativa do algoritmo AMCL.

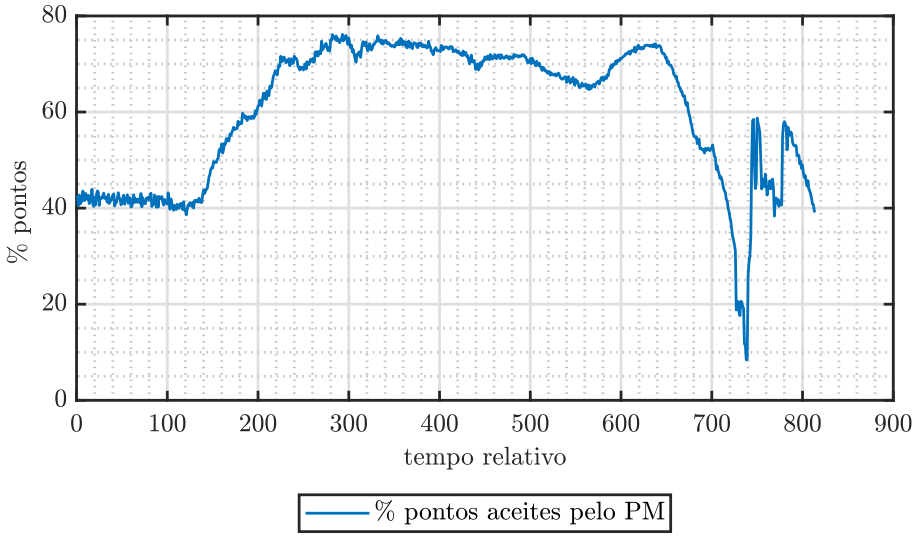


Figura 5.25: Variação da percentagem de pontos aceites pelo algoritmo *Perfect Match* ao longo de todo o percurso do robô.

5.4.1.2 Erros forçados na odometria

Relativamente ao problema detetado no algoritmo AMCL, nomeadamente da grande influência da odometria, o algoritmo desenvolvido por Farias *et al.* [38] demonstrou ser capaz de solucionar este tipo de problema. Ao serem utilizados dois algoritmos de localização, e pelo facto do algoritmo *Perfect Match* ser um algoritmo mais robusto no que diz respeitos aos erros na odometria, o supervisor desenvolvido, consegue prevenir a perda de localização por parte do algoritmo AMCL quando existem fortes erros na odometria, tais como a patinagem ou derrapagem das rodas do robô. De acordo com os resultados obtidos (figura 5.26), quando foram forçados erros na odometria, o algoritmo AMCL começou a divergir da verdadeira posição do robô. No entanto, o algoritmo PM, continuou com uma boa estimativa de localização. Por este facto, e como o supervisor desenvolvido por Farias *et al.* [38] privilegia a pose fornecida pelo *Perfect Match*, quando as diferença entre as poses é superior a um determinado *threshold*, o supervisor reajusta a pose dada pelo AMCL em função do PM, prevenindo assim uma falha no sistema de localização do robô.

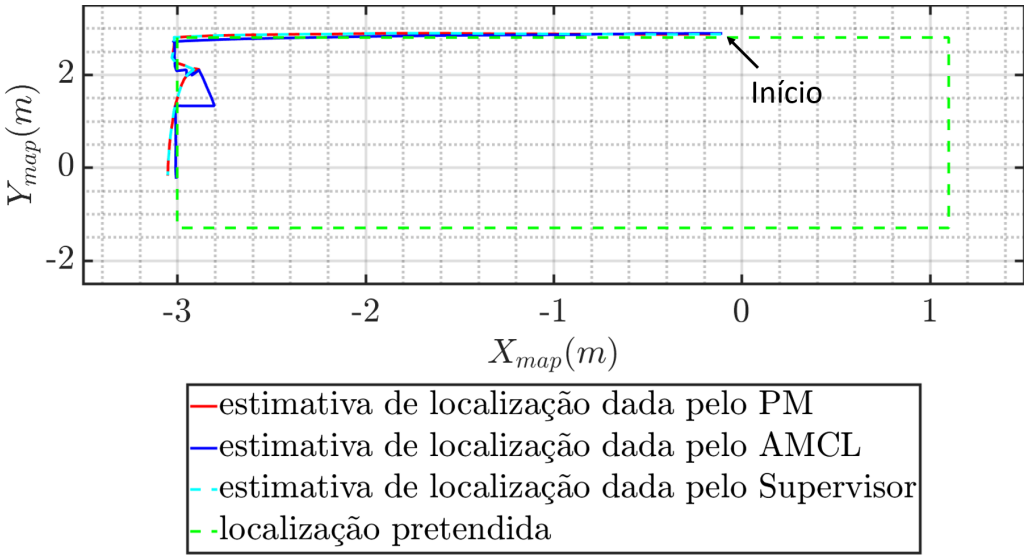


Figura 5.26: Falha na estimativa de localização do algoritmo AMCL provocado por uma má estimativa da odometria, no entanto o supervisor dos autores corrigiu esta falha.

5.5 Mecanismo de deteção de falhas proposto

O mecanismo de deteção de falhas proposto nesta dissertação surgiu com a necessidade de detetar de forma automática situações em que os algoritmos perdem o *tracking* da pose do robô, tais como as mencionadas aquando da análise realizada aos algoritmos de localização e supervisão. O mecanismo desenvolvido tem como objetivo supervisionar o algoritmo de localização Perfect Match. Devido à sua reatividade característica, foi desenvolvido um método de supervisão do

algoritmo que verifica se é ou não possível os "saltos" na estimativa da pose robô resultante do algoritmo, permitindo assim identificar possíveis situações de perda de localização. Deste modo procurou-se dotar o supervisor com o conhecimento quer da localização global, quer do deslocamento relativo permitindo adicionar, ao Perfect Match, uma componente acrescida da odometria.

5.5.1 Conceito geral do supervisor desenvolvido

O supervisor desenvolvido nesta dissertação, tem por base a exploração dos conceitos teóricos abordados até então. Partindo da premissa que o algoritmo de localização global utiliza apenas o deslocamento relativo entre ciclos de controlo, a informação dada pela odometria torna-se mais precisa, permitindo assim a utilização dos dados deste tipo de localização. Desta forma, torna-se viável a sua comparação com os dados oriundos da localização global.

Numa abordagem mais teórica, quando um robô se encontra bem localizado, os dados provenientes da localização global e do deslocamento relativo são coerentes, uma vez que, no referencial do mundo, indicam a mesma posição. Porém, quando existe uma falha de localização (ou na localização global, ou na odometria), estes dados deixam de ser redundantes por um curto período de tempo (uma vez que o método de localização global volta a ajustar a odometria, mesmo que seja de forma errada). Com base neste principio, a ideia geral do supervisor desenvolvido na presente dissertação, passa pela comparação, em cada ciclo de controlo, da posição dada pelo algoritmo de localização global (*Perfect Match*) com os dados provenientes do deslocamento relativo, com o intuito de detetar uma discrepância entre eles. No entanto, a posição fornecida pela localização global não pode ser comparada diretamente com a posição dada pela odometria, uma vez que os dados de localização provenientes dos dois métodos de localização estão expressos em diferentes referenciais. Tal como referido na secção 5.1.1, os dados relativos à odometria, são expressos no referencial **odom**, enquanto que os dados provenientes da localização global encontram-se expressos o referencial **map**. Assim sendo, existe a necessidade de colocar ambas as posições dadas pelos métodos no mesmo referencial para estas poderem ser comparadas e serem extraídas as conclusões necessárias. Contudo, a transformação entre o referencial **map** e o referencial **odom** não é linear em consequência dos erros acumulados na odometria e da correção realizada por parte do mecanismo de localização global. Por conseguinte, existirá sempre um pequeno erro de transformação, isto é, os pontos representados no referencial **odom** não corresponderão exatamente aos pontos no referencial **map**, ainda que este erro seja mínimo quando o robô se encontra bem localizado. Assim sendo, primeiramente o objetivo passa por encontrar a transformação que minimize o erro de *matching* entre os pontos dos referenciais, para assim poderem ser extraídas conclusões.

5.5.2 Aproximação de dois conjuntos de pontos pelo método dos Mínimos Quadrados Recursivos

5.5.2.1 Mínimos quadrados

Com o intuito de identificar a transformação que minimiza o erro de *matching* entre os pontos dos dois referenciais, recorreu-se à técnica matemática denominada por mínimos quadrados. Esta

é uma técnica de otimização matemática, cujo o objetivo é encontrar um estimador que minimize a soma dos quadrados das diferenças entre os valores estimados e os valores realmente observados. No entanto, uma condição necessária para a aplicação deste método, é que a relação entre as variáveis seja linear, bem como o erro que modeliza a diferença entre elas seja ruído branco e de média nula. Matematicamente, para que seja possível aplicar o método dos mínimos quadrados, o modelo do sistema a identificar necessita de estar representado sob a forma:

$$Y = X\hat{\theta} + \varepsilon \tag{5.6}$$

em que $\hat{\theta}$ é a razão de transformação entre a variável de entrada e saída, e ε é a variável que traduz o erro entre as transformações [65] . Portanto, conhecendo as variáveis X e Y e desde que as restrições impostas sejam cumpridas, é possível, através dos mínimos quadrados, estimar o parâmetro $\hat{\theta}$ que minimiza a seguinte expressão matemática:

$$E(\hat{\theta}) = \sum_{i}^{n} (Y_i - X_i\hat{\theta})^2 \tag{5.7}$$

De acordo com [65], para que o estimador resultante dos mínimos quadrados exista e seja único, é necessário $X'X$ seja invertível e que esta seja uma matriz definida positiva (isto é, os seus valores próprios sejam maiores que zero). Caso estas restrições sejam cumpridas, o estimador de mínimos quadrados que minimiza 5.7 é dado por 5.8, sendo a covariância do erro de estimação dado pela equação 5.9.

$$\hat{\theta} = (X'X)^{-1}X'Y \tag{5.8}$$

$$P = (X'X)^{-1} \tag{5.9}$$

5.5.2.2 Mínimos quadrados recursivos

No caso específico desta dissertação, os dois conjuntos de dados são os pontos representados no referencial **map** e no referencial **odom** (Figura 5.27), em que o objetivo é estimar o parâmetro $\hat{\theta}$ de forma a minimizar o erro de *matching* entre os pontos. De uma forma mais teórica e de

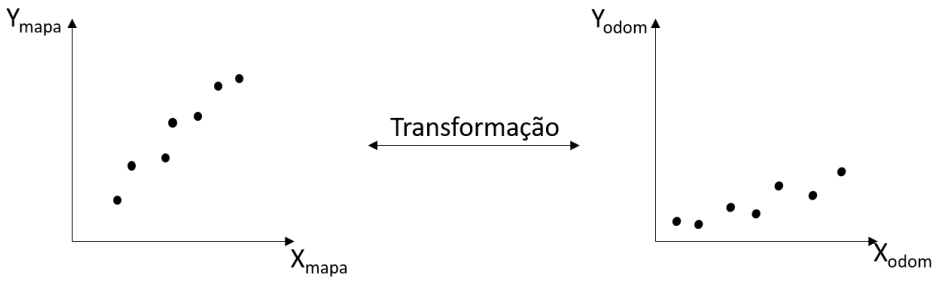


Figura 5.27: Exemplo ilustrativo da transformação necessária a realizar.

acordo com referido na secção 5.1.1, a transformação entre dois referenciais é composta por uma rotação e uma translação, sendo essa relação expressa na equação 5.10.

$$Y = RX + t; \tag{5.10}$$

Exprimindo a equação 5.10 sob a forma matricial (equação 5.11) podemos constatar que esta fica expressa sob a forma dos mínimos quadrados, proporcionando a sua utilização.

$$Y = \begin{bmatrix} R & t \end{bmatrix} \times \begin{bmatrix} X \\ 1 \end{bmatrix} \Leftrightarrow Y = \hat{\theta} X_M \tag{5.11}$$

Assim sendo, seria de esperar que, o estimador devolvido pelo método dos mínimos quadrados seria composto exatamente por uma matriz de rotação e uma matriz de translação. No entanto, tal como referido na secção 5.1.1, a matriz de rotação é uma matriz com características únicas, sendo uma delas o facto de esta ser, obrigatoriamente, unitária. Por este motivo, estimar uma matriz composta por uma matriz de rotação, seria um processo não linear, inviabilizando assim o uso dos mínimos quadrados. Todavia, caso o robô esteja bem localizado, o processo de transformação é invariante no tempo, isto é, a transformação aplicada é constante, tornando assim o processo linear e permitindo o uso dos mínimos quadrados para estimar a matriz de transformação. Contudo, é importante realçar que, ao serem utilizados os mínimos quadrados para identificar qual a transformação entre os dados, esta não será composta pela matriz de rotação e pela translação, mas sim por uma matriz de rotação, de translação e um fator de escalonamento, em virtude da não linearidade característica da matriz de rotação que não é tida em conta pelo algoritmo utilizado [66]. Apesar da matriz de rotação e translação não serem estimadas, o principal objetivo passa por encontrar, apenas, uma razão de transformação entre variáveis de tal modo que seja detetada uma discrepância na transformação indicando assim que poderá ter existido um erro, ou na localização global, ou na odometria.

Como o conjunto de dados a utilizar para a aplicação dos mínimos quadrados são amostrados sequencialmente ao longo do período de atividade do robô (o que implica que estes estejam sempre a crescer) existe a necessidade de incorporar este aspeto na sua aplicação. Nesse sentido torna-se necessária a aplicação dos mínimos quadrados à medida que são adquiridos novos dados. Para tal poderia ter sido utilizada a técnica de "janela móvel", na qual consiste a inserção/remoção de dados a cada período de amostragem. Por outras palavras, esta "janela móvel" teria um comportamento onde os dados mais recentes iriam substituindo os dados mais antigos (quando existe uma nova amostra, a mais antiga é descartada, dando lugar à nova). No entanto, este tipo de abordagem foi descartada pelo facto do tamanho da janela a utilizar (isto é, o número de dados a utilizar em cada interação) ser um parâmetro desconhecido, tendo este de ser configurado manualmente com base numa bateria de experiências, e ainda pelo facto de ser computacionalmente mais pesado (devido à inserção e remoção de dados em cada iteração). Por conseguinte, com o intuito de automatizar todo o processo e evitar o uso de janelas fixas, recorreu-se então aos mínimos quadrados recursivos (também conhecido como RLS proveniente do inglês *Recursive Least Squares*). Esta

técnica matemática é uma extensão dos mínimos quadrados que não necessita de uma amostra de dados fixa, uma vez que esta tem a possibilidade de obter uma estimativa do parâmetro $\hat{\theta}$ à medida que novas observações vão sendo obtidas, calculando-a de uma forma recursiva (ou seja, utilizando a estimativa anterior). Desta forma, todos os dados amostrados têm influência no cálculo do estimador. Na figura 5.28 é apresentado um diagrama onde se encontra representado os *inputs* e *outputs* deste mecanismo. Tal como é possível observar, os mínimos quadrados recursivos necessitam para a nova estimação apenas da estimativa anterior e da sua covariância do erro de estimação, bem como da nova amostra de dados. Uma vez processados os dados de entrada, esta ferramenta retorna uma nova estimativa de $\hat{\theta}$ bem como a covariância do erro de estimação associada à mesma, sendo estes dados utilizados na próxima iteração do algoritmo.

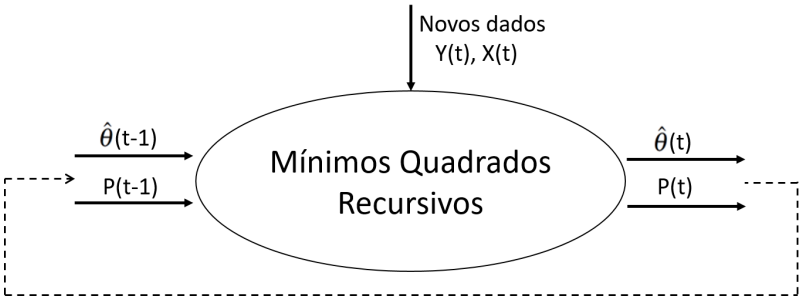


Figura 5.28: Diagrama dos mínimos quadrados recursivos

Através de uma manipulação algébrica e aplicando diversas ferramentas matemáticas às equações 5.8 e 5.9, obtém-se a equação associada ao algoritmo RLS, a qual se encontra representada na equação 5.12, bem como a covariância do erro de estimação associada à mesma (equação 5.14) [65]. De uma forma matemática, a equação para o cálculo do estimador através do algoritmo RLS, é muito semelhante a um filtro de Kalman, em consequência da utilização de um ganho de Kalman (equação 5.13) para a fusão da nova estimativa do parâmetro $\hat{\theta}$ com o anteriormente calculado.

$$\hat{\theta}(t) = \hat{\theta}(t-1) + K(t)[Y(t) - X(t)\hat{\theta}(t-1)] \tag{5.12}$$

$$K(t) = \frac{P(t-1)X(t)}{1 + X(t)'P(t-1)X(t)} \tag{5.13}$$

$$P(t) = [I - K(t)X(t)']P(t-1) \tag{5.14}$$

Utilizando esta ferramenta matemática, apenas é necessário inicializar o algoritmo com uma estimativa inicial da matriz de covariância do erro de estimação. Como o estimador inicialmente é desconhecido, a matriz de covariância é inicializada com valores muito grandes (5.15) de forma a refletir a incerteza associada ao parâmetro a estimar. Consequentemente, numa fase inicial, o ganho de Kalman será grande permitindo um ajuste rápido do estimador para o valor verdadeiro. Com o decorrer do tempo, isto é, à medida que vão sendo adquiridos cada vez mais dados, o ganho de Kalman vai diminuindo em consequência da diminuição de P, levando a que a correção do

parâmetro a estimar seja cada vez mais pequena por estar perto do valor verdadeiro, estabilizando num determinado valor.

$$P(t) = \begin{bmatrix} 1000 & 0 & 0 \\ 0 & 1000 & 0 \\ 0 & 0 & 1000 \end{bmatrix} \tag{5.15}$$

O algoritmo de mínimos quadrados recursivos apresentado até então, pesa igualmente todos os dados, ou seja, pesa de igual forma os dados recentes e os que surgiram num passado remoto. Por conseguinte, caso os dados sejam coerentes durante longos períodos de tempo (no caso concreto desta dissertação, é o facto do robô estar bem localizado durante longos períodos), o ganho de Kalman torna-se muito baixo, tornando o algoritmo muito lento a detetar uma alteração. Com o intuito de ultrapassar esta limitação do algoritmo, recorreu-se à utilização dos mínimos quadrados com fator de esquecimento exponencial [65]. Ao ser utilizada esta vertente do algoritmo, é atribuído um maior peso aos dados mais recentes face aos dados do passado, através da alteração da função custo face ao algoritmo original. Mais precisamente, a alteração consiste na modificação da função custo, acrescentando o parâmetro λ (denominado fator de esquecimento) o qual pode variar entre $0 < \lambda < 1$, correspondendo os valores próximos de 1 à retenção de uma grande quantidade de dados para o cálculo do parâmetro a estimar (ou seja, é dado um peso significativo quer aos dados do presente, quer aos do passado). Já os valores próximos de 0 correspondem à retenção de poucos dados, privilegiando desta forma, apenas os dados adquiridos mais recentemente, ou melhor, é dado pouco peso ao dados menos recentes (equação 5.16).

$$E(\hat{\theta}) = \sum_{i}^{n} \lambda^{n-i} (Y_i - \hat{\theta} X_i)^2 \tag{5.16}$$

Ao ser alterada a função custo do algoritmo, as equações 5.13 e 5.14, são consequentemente alteradas de modo a respeitar a alteração efetuada na função custo:

$$K(t) = \frac{P(t-1)X(t)}{\lambda + X(t)'P(t-1)X(t)} \tag{5.17}$$

$$P(t) = [I - K(t)X(t)'] \frac{P(t-1)}{\lambda} \tag{5.18}$$

A título de exemplo, na figura 5.29, encontram-se dois conjuntos de pontos, sendo que, um deles está expresso no referencial **map** (a azul) e outro no referencial **odom** (a vermelho). Utilizando assim a técnica dos mínimos quadrados recursivos com fator de esquecimento exponencial, é possível estimar o parâmetro $\hat{\theta}$ que melhor se ajusta aos conjuntos de dados, permitindo desta forma converter os pontos expressos no referencial **odom** para o referencial **map**, tal como demonstra a figura 5.30.

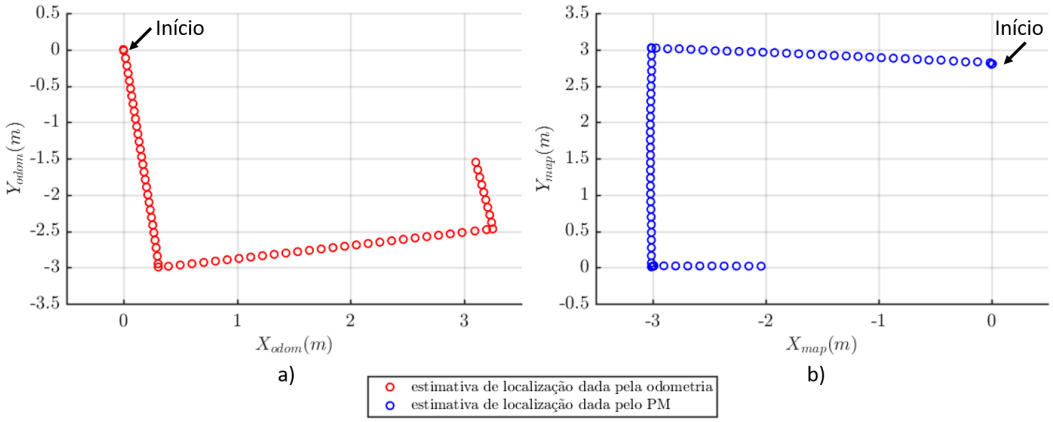


Figura 5.29: Representação de dois conjuntos de pontos expressos em referencias diferentes. a) Representação do conjunto de pontos expresso no referencial **odom**. b) Representação do conjuntos de pontos expresso no referencial **map**.

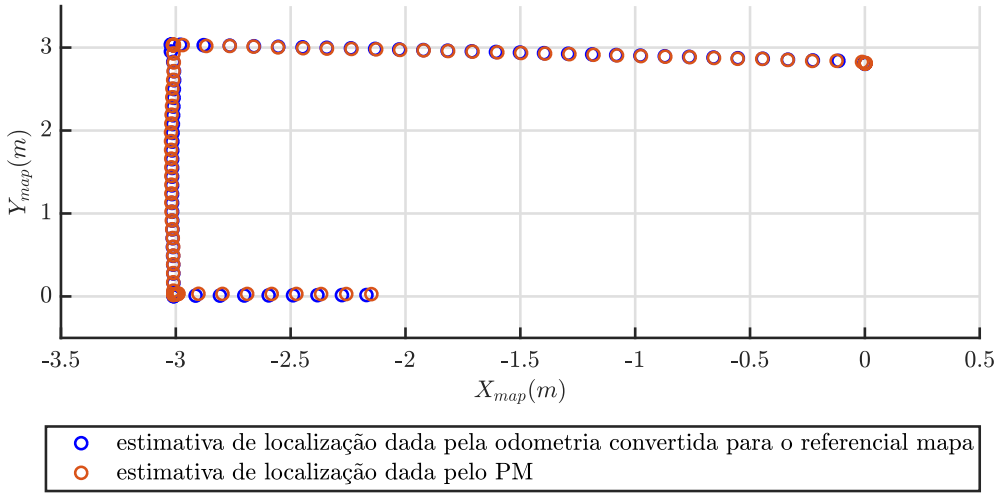


Figura 5.30: Representação dos dois conjuntos de pontos no mesmo referencial (**map**), após estimação do parâmetro $\hat{\theta}$ pelo método dos mínimos quadrados recursivos com fator de esquecimento exponencial.

5.5.3 Mecanismo de deteção de falhas desenvolvido

Tal como abordado na secção 5.5.1, o objetivo principal do mecanismo consiste na comparação direta entre os dados da odometria e os dados provenientes da localização global, com a finalidade de verificar a viabilidade dos "saltos" dados pela estimativa da localização global, nomeadamente, dos "saltos" dados pela estimativa da pose dada pelo algoritmo *Perfect Match*.

Uma vez estabelecido o algoritmo a utilizar para obter a matriz de transformação que minimiza o erro de *matching* entre os pontos dos referenciais, passa a ser possível analisar os dados e assim retirar as respetivas conclusões. Para tal recorreu-se, tal como referido na secção 5.5.2, ao algoritmo de mínimos quadrados recursivos com um fator de esquecimento exponencial o qual

se encontra, sob a forma de pseudocódigo, representado no algoritmo 2. Face ao algoritmo original, foi implementado ainda o cálculo da norma do erro de *matching* (linha 8) como forma de avaliar qual o grau de *mathing* entre os pontos e assim obter um indicador de desempenho dos algoritmos de localização. Por outras palavras, caso a norma do erro se encontre em níveis aceitáveis, isto é, o erro de *matching* entre os dados for baixo, é sinónimo de uma boa localização indicando que os dados de ambos os algoritmos de localização são coerentes. Caso o indicador de desempenho ultrapasse um determinado valor, isto é, acima de um dado limite, é indicador de falta de coerência entre os dados, indicando uma falha num dos algoritmos de localização. Desta forma, desenvolveu-se o algoritmo 3, o qual tem como entradas, quer a localização resultante do algoritmo (*AlgorithmPose*), quer a localização fornecida pela odometria (*odomPose*), e ainda um parâmetro referente ao limite a utilizar. Primeiramente, é chamada a função *RLS* (linha 2), com o intuito de obter a norma do erro associada ao *matching* de dados. Consoante o resultado retornado pelo algoritmo RLS é atribuído o estado dos algoritmos de localização, isto é, se os dados dos algoritmos são coerentes ou não (linhas 3-7). No caso de não serem coerentes, é sinónimo de falha possível falha de localização, isto é, existiu uma falha, ou no algoritmo de localização global, ou na odometria.

Algorithm 2: Pseudocódigo do algoritmo RLS - *Recursive Least Squares*

Input: $odomPose$, $AlgorithmPose$, λ, $\hat{\theta}(t-1)$, $P(t-1)$
Result: $errorNorm$

1 **begin**
2 $X = [odomPose\ 1]$;
3 $Y = AlgorithmPose$;
4 $kalmanGain = \frac{P(t-1)\times X}{\lambda + X' \times P(t-1) \times X}$;
5 $covariance = [I - kalmanGain \times X'] \frac{P(t-1)}{\lambda}$;
6 $\hat{\theta} = \hat{\theta}(t-1) + kalmanGain \times (Y - X \times \hat{\theta}(t-1))$;
7 $estimatedY = X \times \hat{\theta}$;
8 $errorNorm = norm(estimatedY - Y)$;
9 $return\ errorNorm$;
10 **end**

5.5.4 Resultados

Tal como referido na secção 5.5.2.2, caso o fator de esquecimento seja próximo de 1, são tidos em conta a maior parte dos dados adquiridos, ou seja, quer os dados mais recentes, quer os dados do passado, possuem um elevado peso para a estimação do parâmetro $\hat{\theta}$. Em contrapartida, caso seja utilizado um fator de esquecimento próximo de 0, é apenas dada importância aos dados mais recentes, tornando a estimação do parâmetro $\hat{\theta}$ muito reativa, o que não é vantajoso pois, é importante que haja um pico na norma do erro de *matching* para que seja detetada a falha (tal como demonstrado na secção 5.5.3). Portanto, com intuito de utilizar uma grande quantidade de dados

Algorithm 3: Pseudocódigo do mecanismo de deteção de falhas desenvolvido

Input: $odomPose$, $AlgorithmPose$, $errorThreshold$
Result: $AlgorithmStatus$

```
1 begin
2     errorNorm = RLS(odomPose, AlgorithmPose, λ, θ̂(t−1), P(t−1));
3     if (errorNorm > errorThreshold) then
4         AlgorithmStatus = false;
5     else
6         AlgorithmStatus = true;
7     end
8 end
```

para o cálculo do parâmetro associado à transformação entre referenciais, utilizou-se um fator de esquecimento $\lambda = 0.985$. Desta forma, é possível atribuir peso superior a uma grande quantidade de dados, ao mesmo tempo que, é suavizado o efeito dos dados remotos nos cálculos mais recentes do parâmetro $\hat{\theta}$. Mais precisamente, ao ser utilizado um fator de esquecimento $\lambda = 0.985$, permite que os últimos 45 pontos adquiridos tenham um peso superior a 50% para a estimação do parâmetro $\hat{\theta}$, privilegiando desta forma os dados mais recentes, no entanto, conferindo também alguma importância aos dados do passado.

Relativamente ao limite associado ao indicador de desempenho, após a realização de diversos testes, este foi estabilizado em $errorThreshold = 0.06$, o que significa que a norma do erro de *matching* entre os dados não pode variar mais do que seis centímetros, conferindo uma pequena margem de erro, tolerando assim os típicos erros associados aos algoritmos de localização.

Com a finalidade de testar a viabilidade do mecanismo desenvolvido, foi-lhe aplicado uma bateria de testes similares aos que foram aplicados aquando da análise de falhas nos algoritmos de localização e supervisão. Deste forma, torna-se possível verificar se o mecanismo de deteção de falhas desenvolvido responde aos critérios que levaram ao seu desenvolvimento.

5.5.4.1 Ambiente de navegação com *outliers*

Relativamente aos testes aplicados para o teste do mecanismo de deteção de falhas proposto, recorreu-se a testes semelhantes aos aplicados aquando da análise dos algoritmos de localização e supervisão. Como foi demonstrado, quando é obstruída uma zona do mapa e o robô se encontra próxima da mesma, o algoritmo PM falha, ou seja, converge para uma pose errada. Assim, com o objetivo de verificar se o mecanismo proposto deteta esta falha, testou-se o mesmo nestas mesmas condições, encontrando-se os resultados apresentados nas figuras 5.31, 5.32 e 5.33.

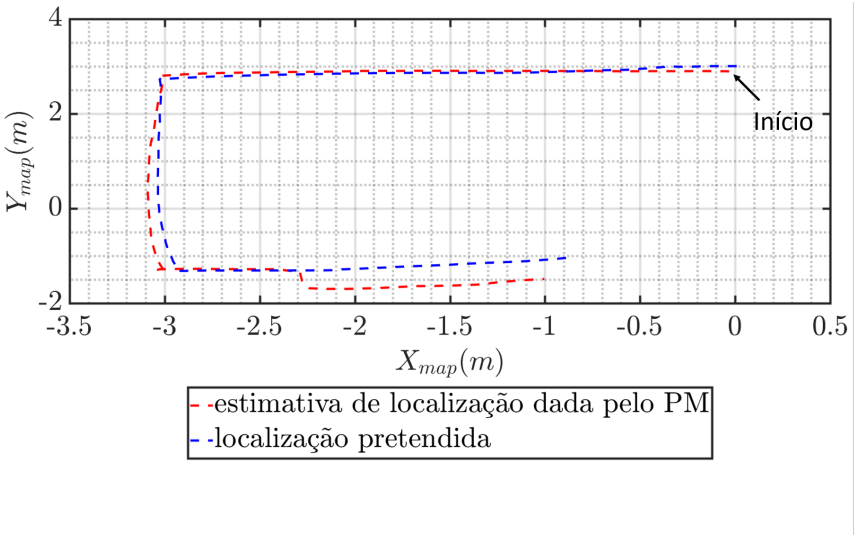


Figura 5.31: Gráfico da estimativa localização dada pelo algoritmo PM, face à localização de referência utilizada, expressa no referencial **map**. No gráfico encontra-se representada uma situação onde o PM diverge da localização pretendida devido à presença de *outliers*.

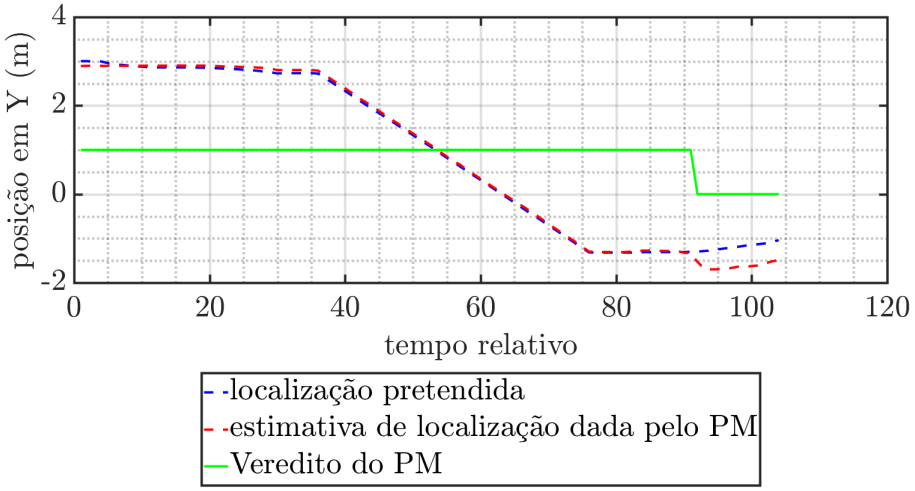


Figura 5.32: Gráfico correspondente à deteção, por parte do mecanismo desenvolvido, da falha de localização do algoritmo PM

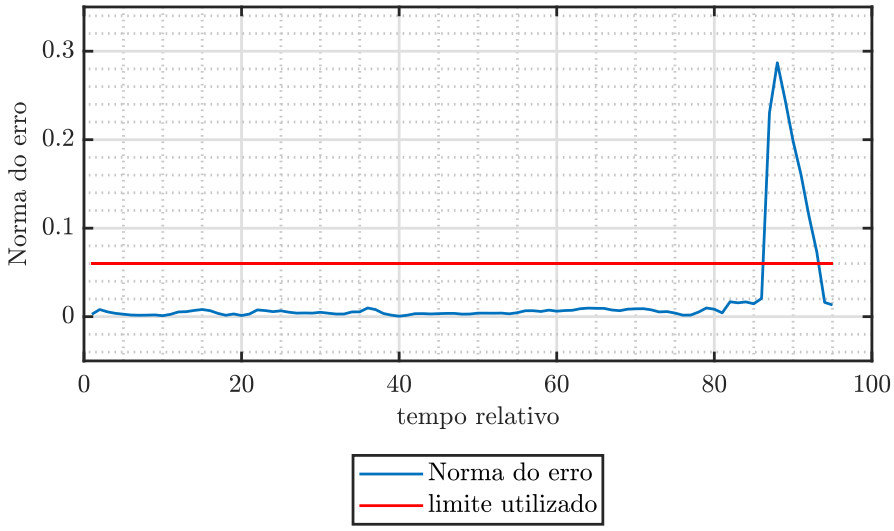


Figura 5.33: Norma do erro de *matching* ao longo de todo o percurso do robô. O pico apresentado corresponde à falha de localização do PM.

5.5.4.2 Erros Forçados na odometria

Como o algoritmo de deteção de falhas verifica o grau de *matching* entre a localização estimada pelo algoritmo PM e a estimada pela odometria, realizaram-se testes onde foram forçados erros na odometria (semelhantes aos aplicados aquando da análise dos algoritmo de localização e supervisão) para verificar o comportamento do mecanismo de deteção de falhas nesta situação, encontrando-se os resultados expressos nas figuras 5.34, 5.35 e 5.36.

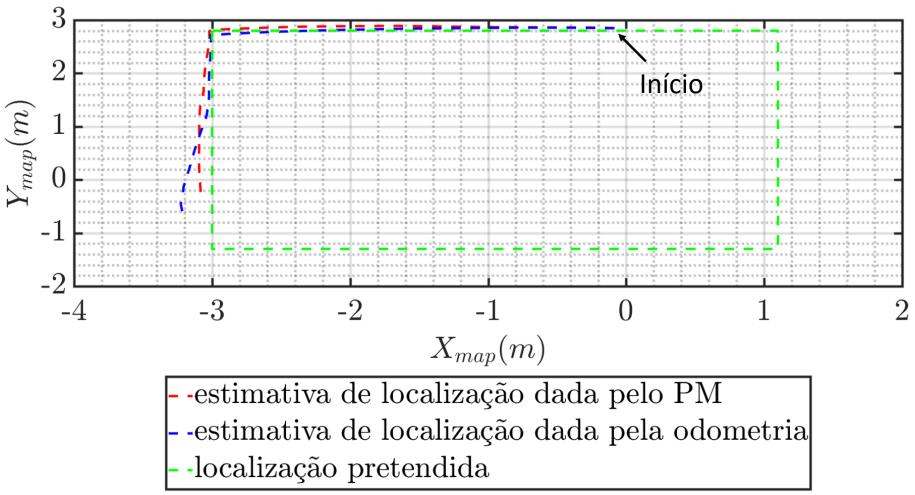


Figura 5.34: Estimativa dada pelo algoritmo PM e da estimativa dada pela odometria face à localização pretendida, expressas no referencial **map**. Na figura está representada a situação onde o PM se encontra bem localizado face a uma má estimativa de deslocamento relativo - indicação de um maior descolamento.

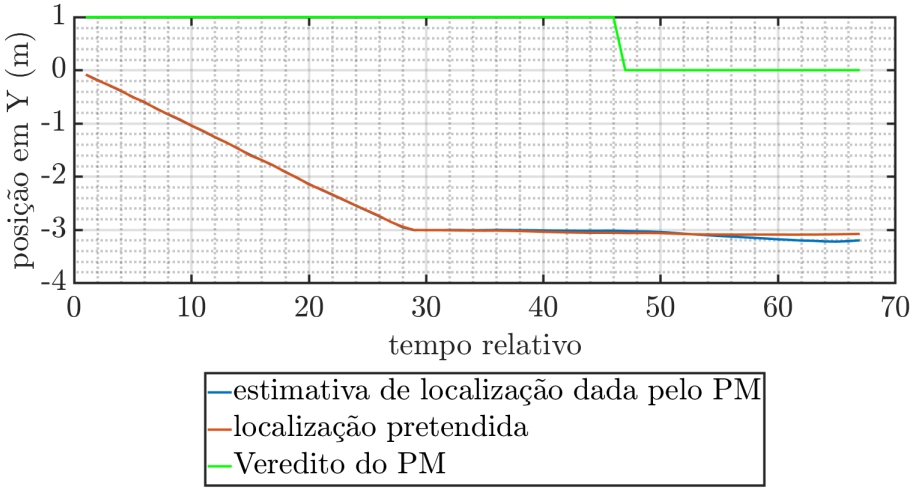


Figura 5.35: Gráfico correspondente à deteção de má localização provocada pelos erros introduzidos na odometria.

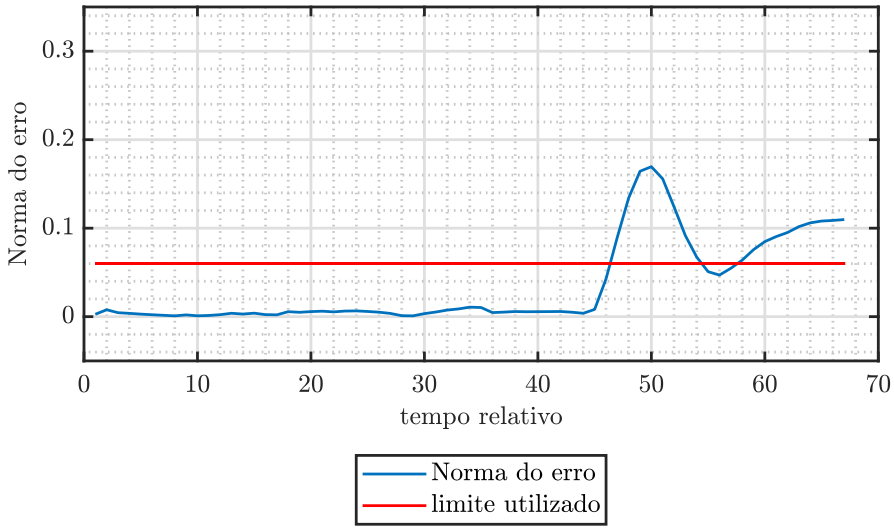


Figura 5.36: Norma do erro de *matching* ao longo de todo o percurso do robô. O pico apresentado corresponde ao erro presente na odometria.

5.5.4.3 Análise dos resultados obtidos

De acordo com os resultados obtidos, e tal como previsto aquando da idealização do mecanismo de deteção de falhas, havendo uma coerência entre os algoritmos de localização, o mecanismo desenvolvido indica que existe uma boa localização do robô. Porém, caso não exista essa mesma redundância, o algoritmo dispara um alerta com o intuito de sinalizar que uma das localizações possui um problema.

O primeiro caso apresentado, isto é, o caso da presença de outliers (secção 5.5.4.1), representa uma má estimativa de localização dada pelo *Perfect Match* na qual o supervisor desenvolvido por Farias *et al.* [38] não era capaz de detetar (figura 5.31). Na figura 5.32 está representada a variação da posição apenas na coordenada Y ao longo do tempo, assim como, o veredito do algoritmo supervisor em termos de falhas de localização (linha a verde, sendo que "1" representa a inexistência de qualquer falha, e "0" caso contrário). Tal como demonstra essa mesma figura, o mecanismo de deteção de falhas desenvolvido deteta a falha de localização, uma vez que o algoritmo de localização global convergiu para uma posição que não é corroborada pela odometria. Como os dados deixam de ser redundantes ao ponto de ultrapassar o *threshold* estabelecido (figura 5.33) o algoritmo de supervisão dispara um alerta, tal como pode ser observado na figura 5.32, através da transição de "1" para "0" representada pela linha a verde.

O mesmo acontece quando existem erros na odometria do robô, tal como demonstram os resultados apresentados na secção 5.5.4.2. Na figura 5.34 está representado um forte erro na odometria, novamente através da patinagem das rodas do robô. Como existiu uma patinagem das rodas, a odometria indica que o robô se movimentou numa dada direção (essencialmente um maior deslocamento), enquanto que o algoritmo de localização global indica que o movimento foi efetuado

noutra (figura 5.34). No presente caso a variação existe na coordenada Y. Tal como é possível observar na figura 5.35, no momento em que os dados provenientes de ambas as localizações começam a divergir, o supervisor dispara novamente o alarme assim que é ultrapassado o *threshold* definido (figura 5.36). No entanto, apesar do erro apresentado pelo deslocamento relativo, o PM continua a estimar corretamente a pose do robô, sendo então considerado este disparo do supervisor como uma falso positivo.

Contudo, consoante testes realizados, um problema deste mecanismo recai no facto de não ser possível distinguir se a falha é proveniente do algoritmo *Perfect Match* ou da odometria, isto é, da odometria. Deste modo, o mecanismo desenvolvido por si só, apenas serve como um fator indicativo de que algo, em termos dos mecanismos de localização, pode estar a falhar. Portanto, para que seja possível identificar a fonte do problema é necessário dotar o supervisor desenvolvido com outros tipos de mecanismos.

5.6 Combinação do método desenvolvido com o supervisor de Farias *et al.* - Supervisor desenvolvido

Tal como verificado na secção anterior, embora o mecanismo de deteção de falhas tenha provado ser capaz de detetar as situações reais de perda da localização do robô por parte do PM, este também demonstrou apresentar falsos positivos. Nomeadamente, quando existem erros na odometria, o mecanismo dispara mesmo que o algoritmo PM continue bem localizado, tornando evidente a falta de capacidade para distinguir se a falha é proveniente da odometria ou se do algoritmo de localização global.

Com o intuito de solucionar a lacuna identificada no mecanismo de deteção de falhas proposto, foram integrados alguns dos conceitos utilizados por Farias *et al.* para o desenvolvimento de um supervisor capaz de resolver os problemas de localização identificados. Desta forma, foi integrada a percentagem de pontos do laser aceites pelo algoritmo PM como fator decisivo para a identificação da origem da falha. Ao contrário do algoritmo de supervisão proposto por Farias *et al.* que utiliza a percentagem de pontos do laser aceites pelo algoritmo PM para avaliar a sua boa localização (que se provou falhar), no presente supervisor é analisada apenas a variação desta percentagem. Tal como é visível na figura 5.25, quando existe uma falha de localização (entre o tempo 700-800) existe uma variação brusca na percentagem de pontos do laser aceites pelo algoritmo. Assim sendo, analisando a variação desta percentagem é possível identificar qual a origem da falha. Por outras palavras, caso exista uma variação brusca na percentagem, é sinónimo de uma falha no algoritmo de localização global do PM. Caso contrário, significa que existe um erro na odometria.

5.6.1 Supervisor resultante da integração dos mecanismos de supervisão

De acordo com as análises realizadas aos algoritmos de localização nas secções anteriores, o algoritmo PM provou ser robusto no que diz respeito às falhas na odometria, sendo mais sensível à presença de outliers. Por outro lado, o algoritmo AMCL provou ser bastante robusto à presença destes, sendo sensível à presença de erros no deslocamento relativo. Com o intuito de manter o *tracking* da posição durante todo o percurso efetuado pelo robô, e uma vez que os algoritmos mencionados demonstram uma complementaridade, o supervisor desenvolvido utiliza em simultâneo, tal como algoritmo de supervisão proposto por Farias *et al.*, estes dois algoritmos de localização global. Desta forma, é possível ter pelo menos um dos algoritmos bem localizado, permitindo assim ao robô navegar sem qualquer problemas de localização desde que sejam detetadas as falhas e sejam tomadas medidas corretivas em conformidade. Da mesma maneira que o algoritmo proposto por Farias *et al.*, quando ambos os algoritmos possuem uma boa localização o algoritmo PM é privilegiado, ou seja, a posição é dada pelo mesmo. Relativamente ao critério para a escolha dos algoritmos este recai no mecanismo de deteção proposto. A figura 5.37 demonstra assim a arquitetura do supervisor proposto.

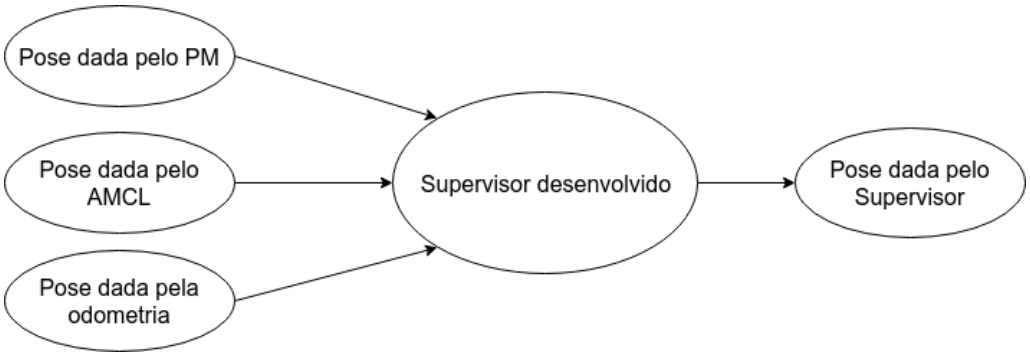


Figura 5.37: Arquitetura do supervisor desenvolvido.

No algoritmo 4 encontra-se representado o pseudocódigo do supervisor desenvolvido tendo este, e tal como referido anteriormente, por base o mecanismo de deteção de falhas proposto na secção anterior. O algoritmo desenvolvido tem como entradas, quer a pose dada odometria (*odomPose*,), quer a pose dada por ambos os algoritmos de localização (*PMpose* e *AMCLpose*). Para além das poses dadas pelos algoritmos de localização, existe ainda um conjuntos de parâmetros associados a *thresholds* utilizados para a tomada de decisão por parte do algoritmo (*errorThreshold*, *laserPointsThreshold*, *ToldistThreshold* e *Cycles2resetPose*). Executado o algoritmo de supervisão, este retorna a pose real do robô - *supervisioPose*.

Relativamente ao algoritmo, este inicia por calcular o estado de cada um dos algoritmos de localização global utilizados (linhas 3 e 4), através do mecanismo de deteção de falhas proposto na secção 5.5.3. Embora o mecanismo de deteção de falhas não tenha sido desenvolvido com o intuito de se aplicar ao algoritmo AMCL (pelo facto de este já possuir uma grande componente relativa à odometria), o mesmo tem a possibilidade de detetar discrepâncias uma vez que, os dados do laser também possuem influência no cálculo da estimativa fornecida pelo algoritmo AMCL,

tornando assim possível detetar igualmente falhas neste mesmo algoritmo, porém não de uma forma tão eficaz. De acordo com o estado dos algoritmos, são tomadas as decisões que levam à escolha da pose real do robô. Caso exista uma falha no algoritmo PM (e o supervisor esteja a utilizar a posição do mesmo), é analisado se esta é oriunda da odometria ou se é do algoritmo em si (linha 6-15). Para tal, é analisada a variação da percentagem de pontos aceites pelo laser (linha 6). Caso esta variação seja superior ao *threshold* definido ($laserPointsThreshold$), é sinónimo de uma má localização no algoritmo PM, sendo a pose escolhida pelo algoritmo de supervisão, a dada pelo algoritmo AMCL (linha 8), reiniciando de seguida o mecanismo de deteção de falhas do algoritmo PM (linha 10). Caso a falha seja proveniente da odometria, a pose escolhida pelo algoritmo de supervisão é a fornecida pelo PM (linha 12). Se a falha detetada estiver relacionada com a odometria, é provável que o algoritmo AMCL comece a falhar devido ao grande peso que esta possui no algoritmo. Desta forma, e como medida preventiva, é corrigida a pose do algoritmo AMCL para a pose dada pelo PM (linha 13), assim como é reinicializado o seu mecanismo de deteção de falhas. No caso do algoritmo PM não possuir nenhuma falha, o supervisor desenvolvido assume a pose real do robô como sendo a fornecida pelo mesmo (linha 18), pelo facto de esta ser mais precisa quando comparada a pose dada pelo outro algoritmo. Relativamente ao algoritmo AMCL, quando este não está a ser utilizado pelo supervisor (isto é, não é o algoritmo que está a fornecer a posição) e o mecanismo de deteção de falhas associado a este deteta uma falha, a pose do algoritmo é reinicializada para a pose dada PM, assim como o seu mecanismo de deteção de falhas (linhas 19-23).

Quando o PM falha, a sua posição não é reajustada com base do algoritmo AMCL uma vez que este poderá voltar a divergir da posição real. Desta forma, quando o algoritmo PM está a falhar, isto é, está a ser utilizada a pose fornecida pelo AMCL, o algoritmo entra num estado especial (linhas 24-42). Mais especificamente, neste estado, o supervisor espera que o PM convirja novamente para uma pose próxima da pose dada pelo AMCL (linha 27-28). Caso o PM não convirja para uma pose relativamente próxima da pose dada pelo AMCL durante um certo número de ciclos (definido pela variável $Cycles2resetPose$), é realizada uma tentativa de reinicialização da pose do mesmo (linhas 32-37). Quando o PM converge, é reinicializado o mecanismo de deteção de falhas associado ao algoritmo (linha 29), passando o supervisor a confiar novamente na estimativa dada pelo PM. Se porventura, no momento em que o algoritmo está a utilizar a posição dada pelo AMCL, receber a indicação, por parte do mecanismo de deteção de falhas, que o mesmo está a falhar, é dada imediatamente uma ordem de paragem ao robô, sendo seguidamente lançado um mecanismo de recuperação da localização do robô (linha 33-34).

Algorithm 4: Pseudocódigo do algoritmo do supervisor desenvolvido

```
Input: odomPose, PMpose, AMCLpose, errorThreshold, laserPointsThreshold,
       ToldistThreshold, Cycles2resetPose
Result: supervisioPose
usingAMCL = false;
begin
    PMstatus = faultDetectionMechanism(odomPose, PMpose, errorThreshold);
    AMCLstatus = faultDetectionMechanism(odomPose, AMCLpose, errorThreshold);
    if (PMstatus == false AND usingAMCL == false) then
        LaserPointsVariation = checkDifferenceLaserPoints();
        if (LaserPointsVariation > laserPointsThreshold) then
            supervisioPose = AMCL_pose;
            usingAMCL = true;
            ResetRLS(PM);
        else
            supervisioPose = PM_pose;
            ResetPose(AMCL);
            ResetRLS(AMCL);
        end
    else
        supervisioPose = PM_pose;
    end
    if (AMCLstatus == false AND usingAMCL == false) then
        supervisioPose = PM_pose;
        ResetPose(AMCL);
        ResetRLS(AMCL);
    end
    if (usingAMCL == true) then
        supervisioPose = AMCL_pose;
        ResetRLS(PM);
        differenceBetweenPoses = abs(PM_pose − AMCL_pose) ;
        if (differenceBetweenPoses < ToldistThreshold) then
            usingAMCL = false;
            ResetRLS(PM);
        else
            cycles++;
            if cycles > Cycles2resetPose then
                ResetPose(PM);
                cycles = 0;
            end
        end
        if (AMCLstatus == false) then
            StopRobot;
            StartGlobalLocatilizationMechanism;
        end
    end
end
```

5.6.2 Resultados

Com a finalidade de validar o supervisor desenvolvido foram-lhe aplicados os mesmos cenários de teste que os aplicados até então. No entanto, para testar a robustez do algoritmo e de modo a simular um situação real de operação do robô, foram introduzidos em simultâneo no percurso do mesmo, quer *outliers*, quer erros forçados na odometria. Desta forma, tonar-se possível, em cada teste aplicado, verificar o funcionamento total do supervisor desenvolvido em termos de deteção de falhas em cada um dos algoritmos e *tracking* da pose do robô.

Relativamente aos parâmetros configuráveis associados ao supervisor, os utilizados para os testes realizados encontram-se representados na tabela 5.4, os quais foram estabelecidos mediante diversos testes, com o intuito de os calibrar para um melhor funcionamento no robô Clever.

Tabela 5.4: Parâmetros utilizados associados ao Supervisor desenvolvido.

errorThreshold	0.06 m
laserPointsThreshold	20.0
ToldistThreshold	0.08 m
Cycles2resetPose	15

5.6.2.1 Ambiente de navegação com *outliers* e com erros forçados na odometria

Tal como referido anteriormente, no presente teste são aplicadas as duas situações de teste dos algoritmos, isto é, no percurso do robô, estão presentes *outliers*, tornando assim possível verificar a falha no algoritmo PM, assim como são forçados erros na odometria com o objetivo de provocar uma falha no algoritmo AMCL.

Portanto, o teste representado na figura 5.38, é composto por dois momentos-chave. O primeiro momento corresponde à primeira parte do percurso do robô, no qual consiste na navegação do robô na presença de *outliers* (figura 5.39a)), e um segundo momento, correspondente à segunda parte do percurso, onde foram forçados erros na odometria (figura 5.39b)). Como é possível observar na figura 5.39a), o algoritmo PM começa a divergir face à localização prevista. No entanto, o supervisor desenvolvido deteta a anomalia no algoritmo e passa a seguir a pose dada pelo algoritmo AMCL. Uma vez que o supervisor desenvolvido dá preferência à pose estimada pelo algoritmo PM, quando o mesmo volta a convergir para uma pose próxima da posição dada pelo algoritmo AMCL, o supervisor volta a seguir a pose dada pelo algoritmo PM. Já na figura 5.39b), é possível verificar uma má estimativa de localização do AMCL, uma vez que este indica um maior deslocamento face à realidade. Por consequência, o supervisor deteta a falha, tomando como ação, a reinicialização da pose do algoritmo para a pose dada pelo PM, impedindo assim que o algoritmo AMCL perca o *tracking* da posição. Na figura 5.40 está representado, ao longo do tempo, qual o algoritmo de localização escolhido pelo supervisor.

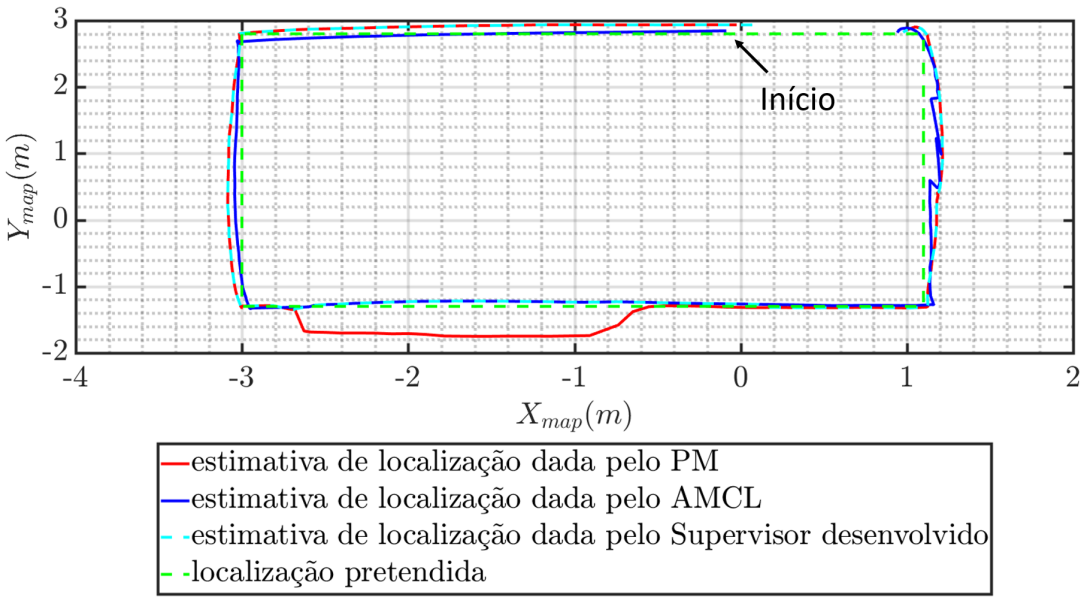


Figura 5.38: Localização dada pelo supervisor desenvolvido face às localizações retornadas pelos algoritmos de localização. Na figura é visível a divergência, quer da estimativa do algoritmo PM, quer da estimativa do AMCL face à localização pretendida, porém o supervisor desenvolvido deteta as situações e reage em conformidade.

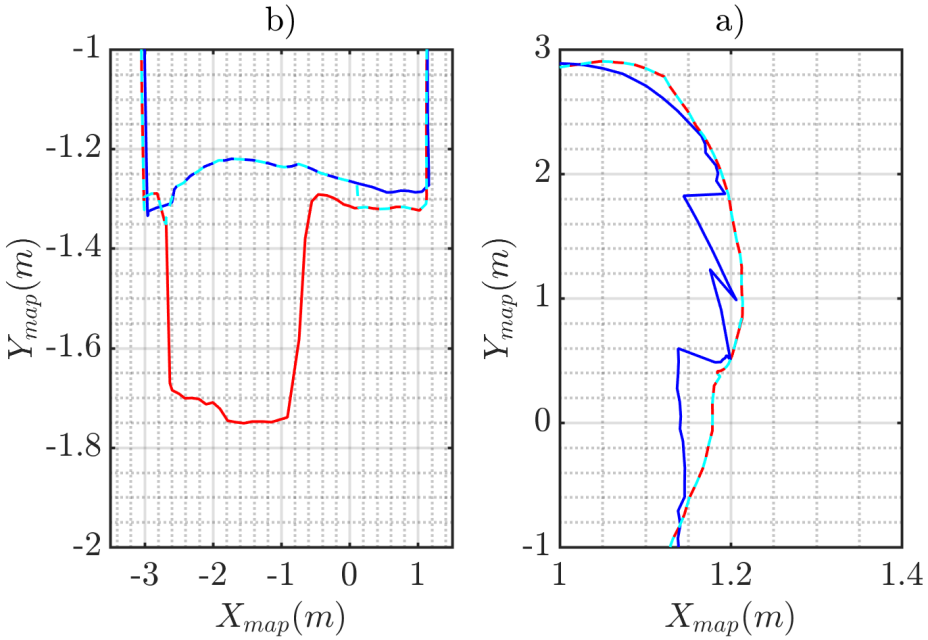


Figura 5.39: Momentos associados às falhas dos algoritmos. Na figura a) está representado a falha de localização associada ao algoritmo PM. Na figura b) está representada a falha de localização no algoritmo AMCL.

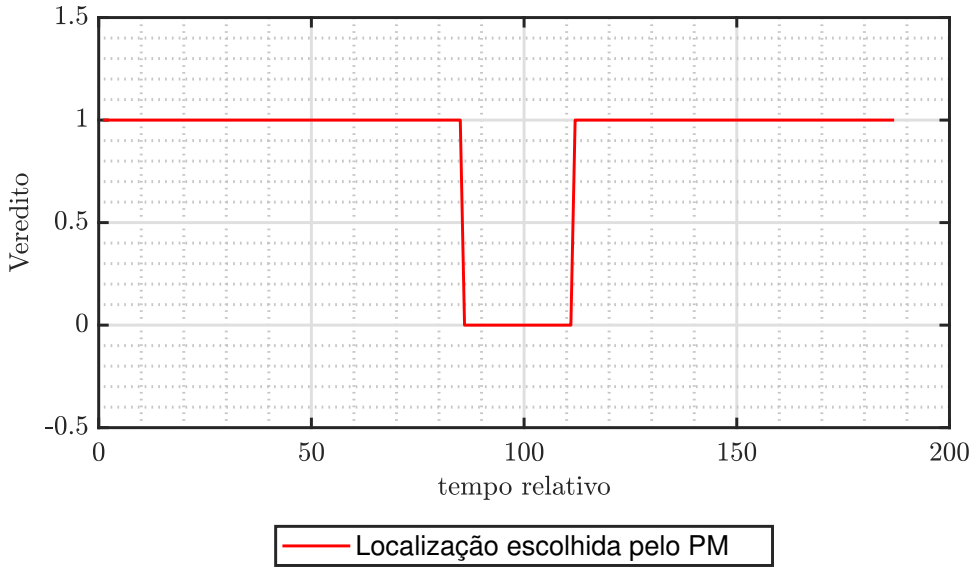


Figura 5.40: Localização escolhida pelo supervisor desenvolvido: 1=PM; 0 = AMCL.

Com o intuito de testar se o supervisor desenvolvido deteta a situação em que ambos algoritmos de localização falham, forçaram-se erros na odometria (nomeadamente, foram elevadas as rodas do robô de modo a simular a patinagem das mesmas) no momento em que algoritmo PM se encontra em situação de falha.

Tal como demonstra a figura 5.41, o algoritmo AMCL acaba por divergir da trajetória prevista, em virtude dos erros introduzidos na odometria. Uma vez que o algoritmo AMCL era o escolhido pelo supervisor (pois o algoritmo PM encontra-se em situação de falha), quando este sofre uma falha, o supervisor deteta a mesma e transita para um estado crítico onde é dada a ordem de paragem do robô e lançado assim um mecanismo de localização global. Na figura 5.42, está representado qual o algoritmo de localização escolhido pelo o algoritmo de localização escolhido ao longo do tempo, sendo possível constatar a identificação da situação crítica referida.

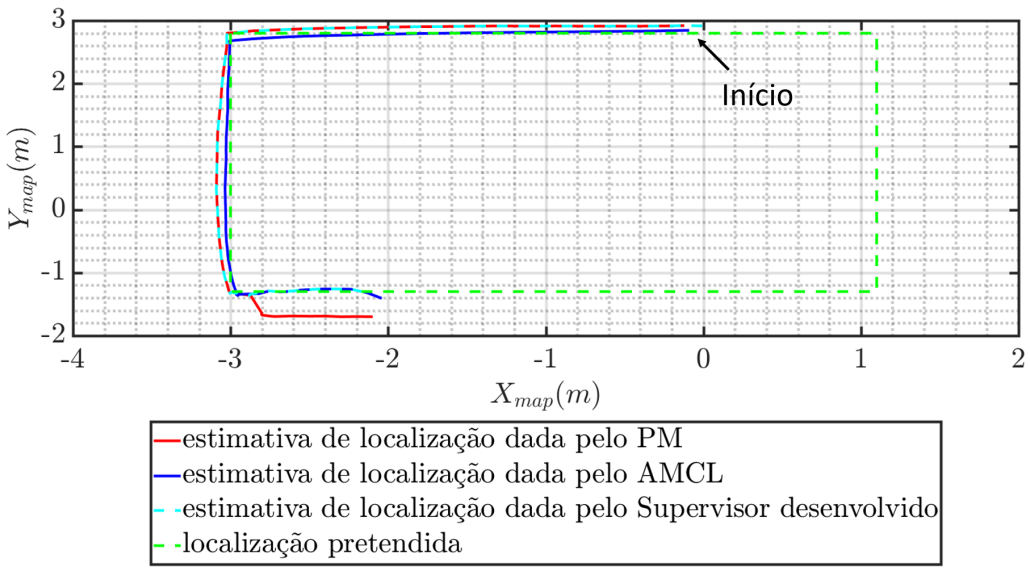


Figura 5.41: Localização dada pelo algoritmo PM, AMCL e pelo supervisor desenvolvido face à localização pretendida. Na figura está representada uma falha, em simultâneo, nos dois algoritmos de localização (AMCL e PM) provocando uma situação crítica de localização.

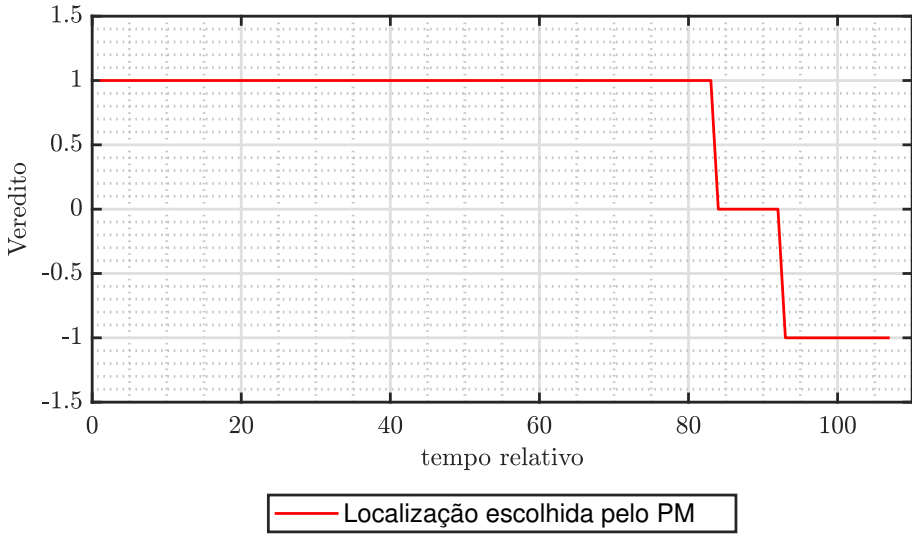


Figura 5.42: Situação crítica detetada pelo supervisor desenvolvido: 1=PM; 0 = AMCL; -1=Erro nos dois algoritmos.

Capítulo 6

Troca automática de baterias

Um dos problemas associados aos robôs móveis, prende-se com a autonomia das baterias utilizadas. Tipicamente, a autonomia das baterias utilizadas está limitada a um determinado número de horas, limitando, por consequência, os períodos de atividade dos robôs. Desta forma, existe a necessidade de recarregar as mesmas, ou até substituí-las. Tal como referido na secção 2.3, no mercado atual, as soluções implementadas, exceto em alguns casos industriais de robôs de grande porte, residem apenas na recarga de baterias. Por conseguinte, os robôs necessitam de suspender as suas atividades para assim poderem recarregar as suas baterias, diminuindo assim a sua rentabilidade. Adicionalmente, existem tarefas, tais como a vigilância, que não toleram períodos "mortos", existindo a necessidade dos robôs estarem operacionais vinte e quatro horas por dia.

Desta forma, com o intuito de eliminar os períodos de inatividade provocados pela recarga de baterias, neste capítulo é proposto um mecanismo de troca automática das mesmas (isto é, sem intervenção humana), tornando assim possível o funcionamento contínuo dos robôs e consequentemente aumentar a sua rentabilidade.

6.1 Sistema de troca de baterias desenvolvido

6.1.1 Sistema de troca automática de baterias proposto

Para que seja possível todo o processo de recarregamento ou troca de baterias é necessário que exista uma *docking station* dedicada para o efeito. No que diz respeito ao tipo de estações, tal como referido na secção 2.3, estas podem ser denominadas ativas ou passivas. As primeiras são caracterizadas por possuírem mecanismos que se movimentam autonomamente com o intuito de auxiliar o robô no processo de docagem. Já as estações passivas, dividem-se em dois tipos: as estáticas e as que possuem graus de liberdade permitindo uma adaptação ao posicionamento do robô com base na força de contacto exercida sobre a estação (tolerando desta forma erros no processo de docagem).

Sendo o custo um fator importante no desenvolvimento do sistema de troca de baterias, optou-se pela desenvolvimento duma estação passiva sem qualquer grau de liberdade. Isso faz com

que toda a complexidade no processo de docagem esteja inerente ao robô, proporcionando uma simplificação na construção da estação.

Por conseguinte, o conceito do sistema de troca de baterias desenvolvido passa pela utilização de duas *docking stations* passivas, dois conjuntos de baterias principais e ainda um conjunto de baterias de *backup* instaladas no robô. No que diz respeito ao processo de troca de baterias proposto, quando o robô necessita de fazer a troca, este dirige-se para a estação que encontra livre, largando a bateria utilizada no local exato para que seja feito o seu recarregamento. Posteriormente, utilizando as suas baterias de *backup*, o robô dirige-se para a outra estação com o intuito de acoplar a outra bateria, que por sua vez já se encontra carregada. Estando completo todo o processo de troca de baterias, o robô encontra-se pronto para continuar com as atividades estipulas para o mesmo.

Como requisito adicional, é necessário que o mecanismo de docagem possua uma elevada precisão, sendo esta, na ordem dos milímetros, devido à estrutura do robô (tal como pode ser constado pela 6.1).

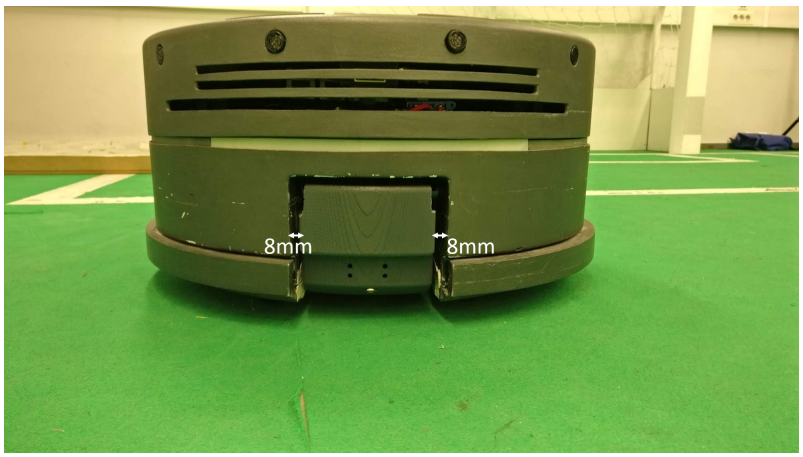


Figura 6.1: Dimensões do espaço livre entre a estrutura do robô e a bateria acoplada ao mesmo.

6.1.2 Estrutura da estação

Tal como foi referido na secção anterior, as *docking stations* projetadas são passivas. No entanto, com o intuito das mesmas auxiliarem o processo de docagem do robô estas foram projetadas com uma geometria específica (figura 6.2.a)). Nomeadamente, as paredes laterais das estações encontram-se com um ângulo de 105° face à parede frontal da estação (figura 6.2.b)). Este ângulo permite conferir uma robustez na deteção da *docking station*, ao mesmo tempo que é minimizado o espaço ocupado pela mesma. Por conseguinte, ao ser utilizado esta tipo de forma geométrica e com estas características, torna possível a identificação de todas as paredes constituintes da estação, facilitando desta forma, não só o processo de deteção da estação, como também o processo de docagem (como será demonstrado na secção 6.1.3). No que diz respeito às dimensões da estação, estas foram estipuladas de acordo com as dimensões do robô Clever, estando as mesmas representadas na figura 6.2.b).

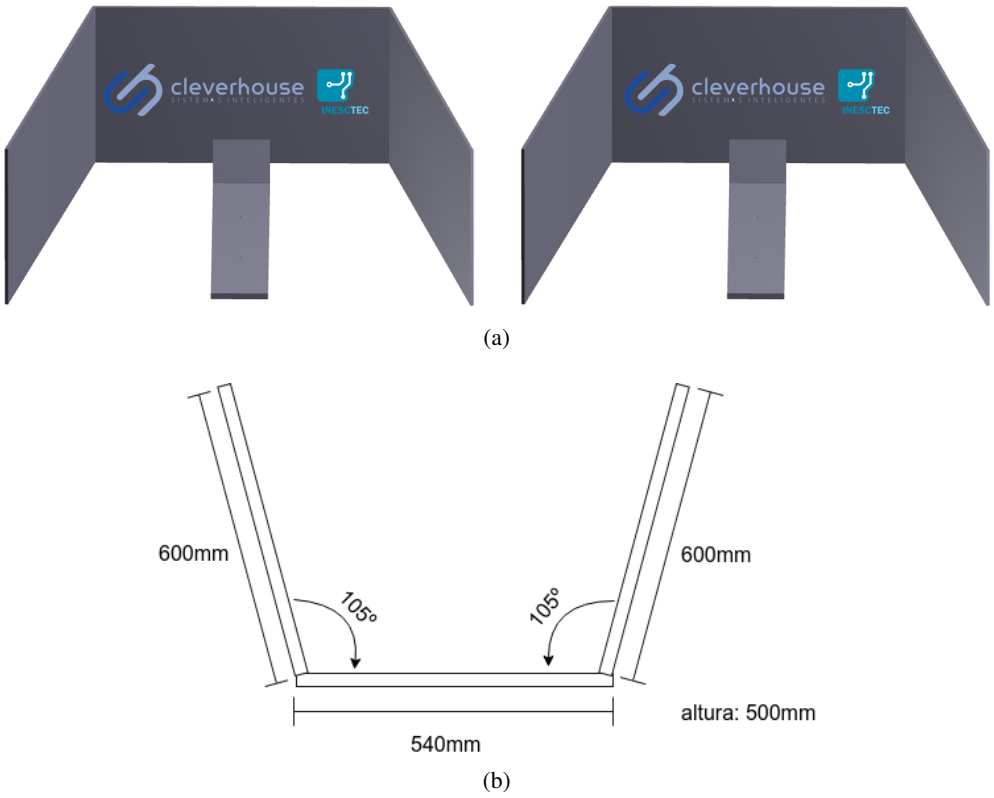
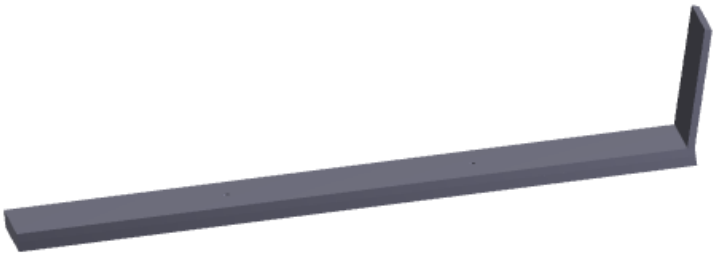


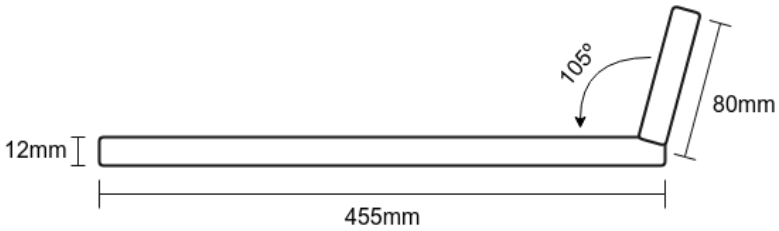
Figura 6.2: Projeto da estação de troca de baterias: a) vista frontal; b) dimensões da estação.

Conforme mencionado no capítulo 4, o robô Clever possui quatro servo motores dedicados para a troca de baterias. No entanto, de acordo com as experiências realizadas, estes demonstraram que não eram capazes de elevar e acoplar a bateria, caso esta se encontre colocada ao nível do solo. Porém, de acordo com os testes realizados, caso a bateria esteja uma distância de 12mm do solo, os servos já conseguem realizar a elevação e o acoplamento da mesma com sucesso. Dada esta característica dos servosmotores, e dado que será necessário ter algo por baixo da bateria para a recarregar, a estrutura desenvolvida (apresentada na figura 6.3), possui uma espessura de 12mm. Para além desta característica, esta possui também dois contactos de cobre, os quais foram pensados para o processo de recarga da bateria que é deixada pelo robô. Portanto, quando o robô larga a bateria na estação, os contactos de cobre presentes na parte inferior da bateria entram em contacto com os da estação, iniciando-se desta forma o processo de recarga da bateria. Adicionalmente, devido à geometria da estrutura desenvolvida, esta permite ainda prevenir um mau posicionamento ou deslocamento na bateria aquando da entrada do robô na estação (caso este possua um pequeno erro de docagem).

O resultado final da estação de recarga desenvolvida encontra-se representado na figura 6.4. Foi através da estação apresentada que foi desenvolvido todo o mecanismo de deteção da estação e todo o mecanismo de docagem.



(a)



(b)

Figura 6.3: Peça de recarregamento projetada: a) vista lateral; b) dimensões e características da peça projetada.

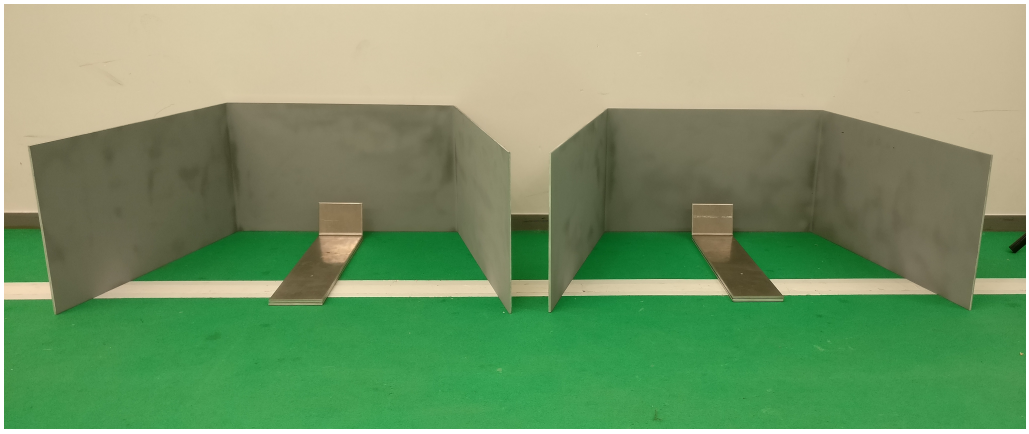


Figura 6.4: Estação real utilizada - vista frontal.

6.1.3 Mecanismo de deteção da *docking station* e trajetória a seguir

O mecanismo de deteção da *docking station* desenvolvido na presente dissertação, consiste não só na deteção da estação, como também na definição da trajetória que o robô terá de seguir para executar uma docagem com sucesso. Adicionalmente este mecanismo determina a posição do robô em relação à estação (ou seja, no referencial da estação) permitindo assim obter-se uma posição para seguir toda a trajetória definida.

Na figura 6.5 é apresentado um exemplo alusivo aos referencias usados no mecanismo proposto. A situação apresentada, embora estejam representados dois robôs distintos ($R1$ e $R2$), tem

como objetivo ilustrar o movimento de um único robô e como as coordenadas dos pontos podem variar consoante o seu movimento. Analisando a figura 6.5, podemos constatar que o ponto P pode ser representado em três referenciais distintos. Mais especificamente, este ponto pode ser representado no referencial da estação $(X_{\text{estação}}, Y_{\text{estação}})$, no referencial associado ao laser presente no robô 1 $(X_{LaserR1}, Y_{LaserR1})$ e no referencial associado ao laser presente no robô 2 $(X_{LaserR2}, Y_{LaserR2})$. De acordo com a figura, o ponto P, no referencial $LaserR1$, possui as coordenadas $(5,-2)$. Já em relação ao referencial $LaserR2$, o mesmo ponto apresenta as coordenadas $(4,1)$. Embora o mesmo ponto possua diferentes coordenadas, quando representados nos referenciais associados aos lasers de cada robô, este ponto possui apenas um par de coordenadas quando representado no referencial da estação, sendo as suas coordenadas $(-5,0)$. Por outras palavras, independentemente da posição dos robôs, como a estação é uma estrutura imóvel, o mesmo ponto possui sempre as mesmas coordenadas, desde que este seja um ponto fixo. Em oposição, o mesmo não se pode inferir quando os pontos são representados em referencias móveis (como é o caso dos referenciais $LaserR1$ e $LaserR2$) pois, mesmo que os pontos sejam fixos, as suas coordenadas podem-se alterar como consequência do movimento do robô. Desta forma, torna-se essencial definir um referencial no qual se possa representar todos os pontos de um forma única e invariável. Portanto, a trajetória que o robô terá de seguir, bem como a sua pose, será representada no referencial da estação, pelas razões anteriormente mencionadas. Outra vantagem de se recorrer ao referencial da estação é o facto da trajetória a seguir passa a corresponder à reta $y = 0$.

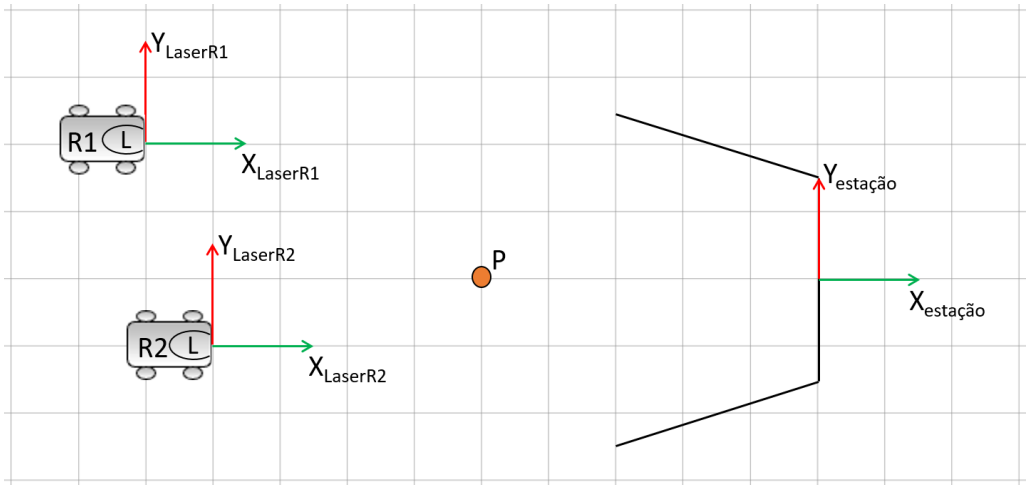


Figura 6.5: Exemplo alusivo ao uso de diferentes referenciais. R1-robô 1; R2-robô 2; L-Laser

No que diz respeito ao algoritmo desenvolvido para a deteção da estação, trajetória a seguir e posição do robô no referencial da estação, o mesmo encontra-se descrito no algoritmo 5. O algoritmo desenvolvido possui como entradas três parâmetros: o primeiro é alusivo à informação obtida pelo *laser* presente no robô (*LaserScanData*); o segundo está associado a um filtro aplicado aos dados provenientes do laser (*MinimumDistance*) e um último intrínseco ao método de identificação da estação. Como *output*, o algoritmo retorna a pose do robô no referencial da estação, estando inerente a esta pose qual a trajetória que o robô necessita de seguir.

Algorithm 5: Pseudocódigo do algoritmo de deteção da estação de recarga

Input: *LaserScanData*, *MinimumDistance*, *threshold*
Result: *robotPose*, *LineToFollow*

```
1 begin
2     filteredLaserScanData = filterData(LaserScanData, MinimumDistance);
3     filteredCoordinates = LaserScanMeasuresToCoordinates(filteredLaserScanData);
4     clusters = findClusters(filteredCoordinates);
5     stationPoints = findStationInClusters(clusters);
6     [leftLine frontalLine rightLine] = findLines(stationPoints);
7     [intersections  stationCenter] = findIntersections(leftLine, frontalLine, rightLine);
8     [X Y θ] = findRobotCoordinates(stationCenter, frontalLine, intersections);
9 end
```

Tal como referido no capítulo 4, o *laser* utilizado no robô Clever (que se encontra invertido) possui uma área de deteção de 240°, num intervalo [-120°; 120°]. No entanto, este ângulo de deteção é alcançado através de incrementos de 0.36°, possuindo, para cada incremento, uma medida de distância associada. Cada uma dessas medidas representa um ponto detetado. Porém, como a estrutura do robô limita o campo de visão do laser, primeiramente são filtradas as medidas detetadas pelo mesmo, sendo descartadas as medidas inferiores a um determinado limite de distância (*MinimumDistance*) (linha 2). Devido à necessidade de representar os pontos detetados no espaço e tendo em conta a posição do laser, nomeadamente a sua inversão, o algoritmo desenvolvido converte os pontos filtrados para coordenadas cartesianas e expressos no referencial do laser através das equações 6.1 (linha 3). O resultado das primeiras duas linhas do algoritmo encontra-se representado na figura 6.6, onde é possível verificar a representação dos dados obtidos pelo laser (já filtrados) convertidos em coordenadas cartesianas.

$$\begin{cases} X_i = Range_i * cos(Angle_i) \\ Y_i = -Range_i * sin(Angle_i) \end{cases} \tag{6.1}$$

Uma vez extrapoladas as coordenadas dos pontos detetados para o referencial do laser, é aplicada uma heurística para agrupar os pontos detetados em *clusters*, isto é, em grupos de pontos que pertençam à mesma estrutura (linha 4). Mais precisamente, a heurística utilizada passa pela análise da distância euclidiana entre pontos consecutivos. Caso esta seja superior a um determinado limite (*threshold*), é considerado como um *breakpoint* entre dois conjuntos de pontos, ou seja, entre duas possíveis estruturas (figura 6.7). Como o robô se encontrará relativamente perto da estação e tendo em conta o efeito radial do laser (ou seja, objetos mais próximos possuem um maior número de pontos que os mais distantes), é escolhido o *cluster* com maior número de pontos, pois teoricamente este será o *cluster* associado à estação devido à sua proximidade em relação ao robô (linha 5). Na figura 6.8 é apresentada a estação identificada no conjunto de pontos detetados pelo laser.

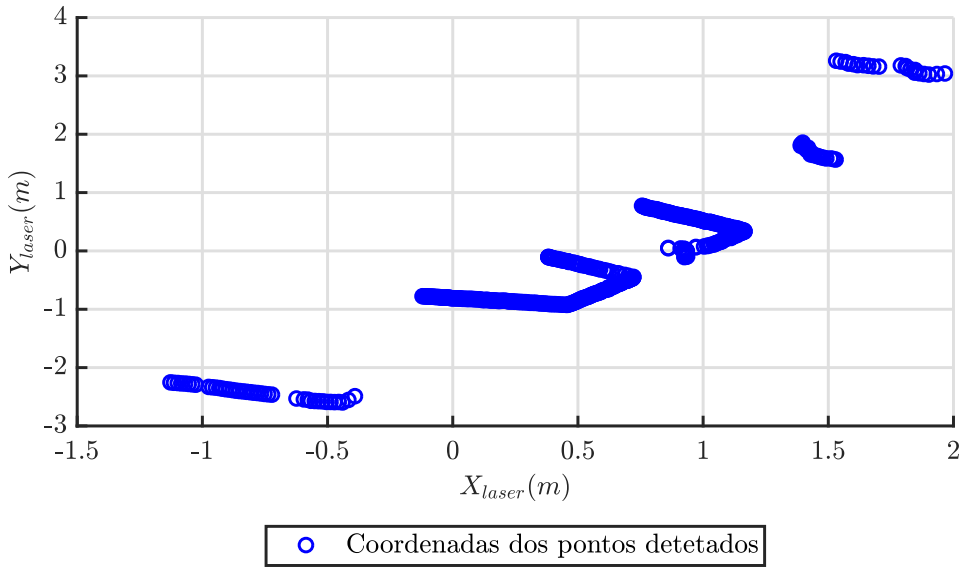


Figura 6.6: Medidas do laser já filtradas e convertidas em coordenadas no referencial do laser - a coordenada (0,0) é onde se encontra o laser.

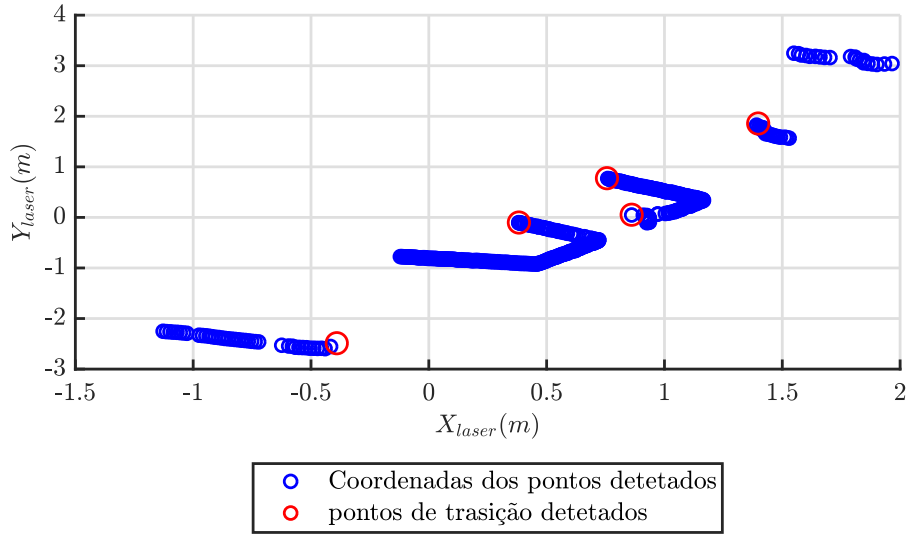


Figura 6.7: *Breakpoints* detetados (identificados a vermelho) associados aos conjuntos de pontos detetados pelo laser.

Identificada e isolada a estrutura da estação de troca de baterias, o próximo passo do algoritmo desenvolvido passa pela identificação das paredes constituintes da estação (linha 6). Para tal, recorre-se à aproximação do conjuntos de pontos da estação por três retas, identificando assim a parede lateral direita, a parede frontal e a parede lateral esquerda da estação. Uma reta pode ser

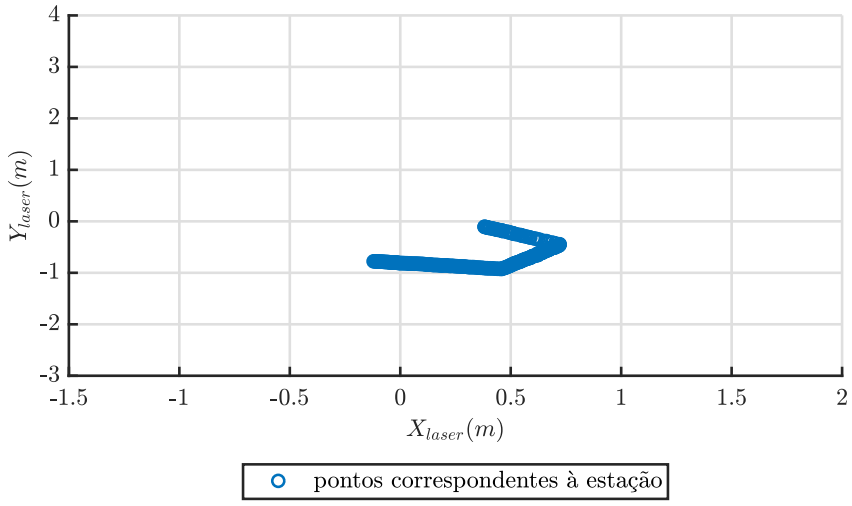


Figura 6.8: Estação identificada e isolada no conjunto de pontos detetados.

aproximada por um polinómio de primeira ordem, como por exemplo pelo polinómio:

$$Y_i = mx_i + b \tag{6.2}$$

sendo m o declive da reta e b o ponto de interseção da reta com o eixo das ordenadas. Uma vez que os dados são conhecidos (Y e x), torna-se necessário encontrar m e b que melhor se ajustam às paredes da estação. Portanto, para o cálculo do declive e do ponto de interseção com o eixo das ordenadas de cada uma dessas retas, recorreu-se a um método de regressão linear, nomeadamente ao método dos mínimos quadrados [67]. Ao ser utilizado este método, o mesmo devolve m e b, que melhor se ajustam a um conjuntos de pontos, utilizando como função custo o erro quadrático, tal como apresentado na equação 6.4.

$$e_i = Y_i - (mx_i + b) \tag{6.3}$$

$$S = e_1^2 + e_2^2 + ...e_n^2 = \sum_{i=1}^{N} e_i^2 = \sum_{i=1}^{N} (Y_i - (mx_i + b))^2 \tag{6.4}$$

Como o objetivo dos mínimos quadrados é minimizar S, isto é, minimizar a soma do erro quadrático, os coeficientes m e b são determinados através do cálculo das derivadas parciais de S em função dos parâmetros a identificar, igualando o resultado a zero (equações 6.5 e 6.6).

$$\frac{\partial S}{\partial m} = -2\sum_{i=1}^{N} x_i(Y_i - (mx_i + b)) = 0 \Leftrightarrow \sum_{i=1}^{N} x_i(Y_i - (mx_i + b)) = 0 \tag{6.5}$$

$$\frac{\partial S}{\partial b} = -2\sum_{i=1}^{N} (Y_i - (mx_i + b)) = 0 \Leftrightarrow \sum_{i=1}^{N} (Y_i - (mx_i + b)) = 0 \tag{6.6}$$

Desta forma, torna-se possível formular o sistema de equações 6.7.

$$\begin{cases} \sum_{i=1}^{N} x_i(Y_i - (mx_i + b)) = 0 \\ \sum_{i=1}^{N} (Y_i - (mx_i + b)) = 0 \end{cases} \Leftrightarrow \begin{cases} m\sum_{i=1}^{N} x_i^2 + b\sum_{i=1}^{N} x_i = \sum_{i=1}^{N} x_i Y_i \\ m\sum_{i=1}^{N} x_i + Nb = \sum_{i=1}^{N} Y_i \end{cases} \tag{6.7}$$

Resolvendo o sistemas de equações apresentado em 6.7, obtém-se que:

$$m = \frac{N\sum_{i=1}^{N} x_i Y_i - \sum_{i=1}^{N} x_i \sum_{i=1}^{N} Y_i}{N\sum_{i=1}^{N} x_i^2 - (\sum_{i=1}^{N} x_i)^2} \tag{6.8}$$

$$b = \frac{\sum_{i=1}^{N} Y_i - m\sum_{i=1}^{N} x_i}{N} \tag{6.9}$$

Portanto, a função *findLines* utilizada na linha 6 do algoritmo 5, recorre às equações 6.8 e 6.9 para calcular quais os parâmetros de cada reta associada a cada parede da estação. Como a estrutura da estação é contínua torna-se necessário identificar os pontos de transição entre as paredes da estação. Para tal, o algoritmo analisa o erro quadrático S (equação 6.4) para identificar novas retas. Por outras palavras, quando o erro é superior a um determinado limite é sinónimo de que o ponto analisado pertence a uma reta diferente da que está a ser calculada, transitando assim o algoritmo para a identificação da nova parede da estação. Porém, quando é detetada uma transição entre paredes da estação, são descartados os últimos 10 pontos (sendo este valor, determinado experimentalmente), com o intuito de obter melhores resultados na estimação da reta correspondente à estação. Seguidamente são detetados os pontos de interseção das retas respeitantes às paredes laterais com a reta associada à parede frontal da estação, com o intuito de identificar o comprimento do segmento de reta associado à parede frontal. Desta forma, torna-se possível identificar o ponto central da estação (linha 7). O resultado após a execução das linhas 6 e 7 do algoritmo encontra-se ilustrado na figura 6.9.

Tal como referido no inicio da secção, de forma a representar a pose do robô de forma única e independentemente da posição do robô, a mesma será representada no referencial da estação. Por conseguinte, a última linha do algoritmo corresponde à estimação da posição do robô expressa no referencial da estação. Para tal a técnica usada para a estimar a pose do robô, consiste na estimação da matriz de rotação e de translação que converte os pontos representados no referencial do laser (ou seja, todos os pontos apresentados até então) no referencial da estação, minimizando o erro quadrático de *matching*. Por outras palavras, o objetivo é encontrar a matriz de rotação R e o vetor translação t que minimize a equação:

$$(R,t) = min\sum_{i=1}^{n} ||(Rp_i + t) - q_i||^2 \tag{6.10}$$

onde, p_i e q_i correspondem aos pontos expressos no referencial do laser e da estação, respetivamente.

No que diz respeito à estimação da matriz R e t, a técnica utilizada é a descrita no artigo [68]. De uma forma sucinta, a técnica utilizada pelo autor encontra-se representada, sob a forma de

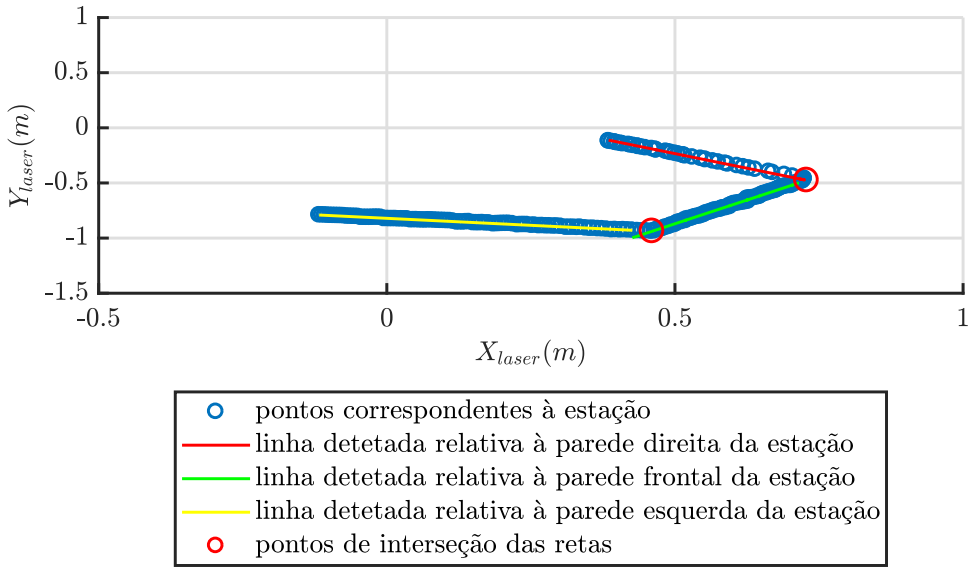


Figura 6.9: Linhas associadas às paredes da estação identificadas, bem como a interseção das mesmas.

pseudocódigo, no algoritmo 6.

Algorithm 6: Pseudocódigo do algoritmo de deteção da estação de recarga

Input: q, p
Result: R, t
begin
  $\bar{p} = \frac{\sum_{i=1}^{n} p_i}{n}$;
  $\bar{q} = \frac{\sum_{i=1}^{n} q_i}{n}$;
  $X = p - \bar{p}$;
  $Y = p - \bar{q}$;
  $CovMatrix = XY^T$;
  $[U \;\; S \;\; V] = svd(CovMatrix)$;
  $R = V \begin{bmatrix} 1 & 0 \\ 0 & det(VU^T) \end{bmatrix} U^T$
  $t = \bar{q} - R\bar{p}$
end

Tal como é possível observar no algoritmo 6, este possui como entradas dois conjuntos de pontos: p e q, sendo estes dois conjuntos expressos em referenciais diferentes, tendo como *output*, a matriz de rotação e o vetor de translação que minimizam a equação 6.10. No caso específico desta dissertação, o conjunto de pontos utilizados representam as interseções das paredes laterais com a frontal (*intersections*), e ainda o ponto central da estação (*stationCenter*), expressos quer

no referencial do laser, quer no referencial da estação. Ao serem utilizados apenas estes pontos, os resultados provenientes do mecanismo proposto tornam-se independentes do grau de precisão da estrutura da estação, ou seja, imune a variações na geometria da estação. De acordo com os resultados obtidos (para este conjunto de pontos), não houve a necessidade de aumentar o número de pontos para a estimação da matriz de rotação e translação entre referenciais. No entanto, os pontos referidos até então estão representados apenas no referencial do laser, pelo que é necessário converter os pontos para o referencial da estação. Porém, e tal como se pode constatar pela figura 6.5, como o eixo das ordenadas do referencial da estação é colinear com a reta associada à parede frontal da estação, e o eixo das abcissas é colinear com a mediatriz da mesma, os pontos são facilmente convertidos para o referencial da estação utilizando a seguinte fórmula:

$$\begin{cases} x_{re} = 0 \\ y_{re} = \frac{x_{rl} - b_{\text{estação}}}{m_{\text{estação}}} - y_{\text{pontoCentralEstação}} \end{cases} \tag{6.11}$$

onde *re*, *rl*, representam, respetivamente, o referencial da estação e o referencial do laser.

No que diz respeito ao algoritmo em si, este começa por calcular e retirar a média aos dois conjuntos de pontos, com o objetivo de remover a componente de translação aos pontos em questão para poder ser analisada, exclusivamente, a rotação aplicada ao conjuntos de pontos (linhas 1-5). Uma vez retirada a translação aos conjuntos de pontos, é calculada a, denominada pelos autores [68], matriz de "covariância" (linha 6), com o intuito de calcular os seus valores singulares (linha 7). Através dos valores singulares é calculada a matriz de rotação (linha 8), culminando com o cálculo do vetor de translação, utilizando para tal, a matriz de rotação calculada e os valores médios do conjunto de pontos (linha 9).

Uma vez identificada a matriz de rotação R e o vetor de translação t, é possível converter qualquer ponto expresso no referencial do laser, no referencial da estação utilizando a equação 6.12, tornando-se assim possível de estimar a pose do robô neste mesmo referencial.

$$q = Rp + t \tag{6.12}$$

No entanto, a equação 6.12 apenas permite extrair a posição do robô e não a sua orientação. Porém a orientação do robô encontra-se inerente à matriz de rotação R estimada. Mais precisamente, a matriz de rotação associada a referenciais 2D pode ser representada por:

$$R = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \tag{6.13}$$

Desta forma, torna-se possível obter θ. Como, para ângulos entre $[-90;90]$ o cosseno não fornece informação sobre a direção da rotação (devido às suas propriedades trigonométricas), para esse efeito foi utilizada a informação dada pelo seno. Portanto, a orientação do robô é obtida através da equação 6.14.

$$\theta_{\text{robô}} = \arcsin(c) \tag{6.14}$$

Na figura 6.10, está ilustrado o resultado final do algoritmo, onde se encontra representada a pose estimada do robô (posição e orientação) bem como a linha que o robô tem de seguir (ambas expressas no referencial da estação).

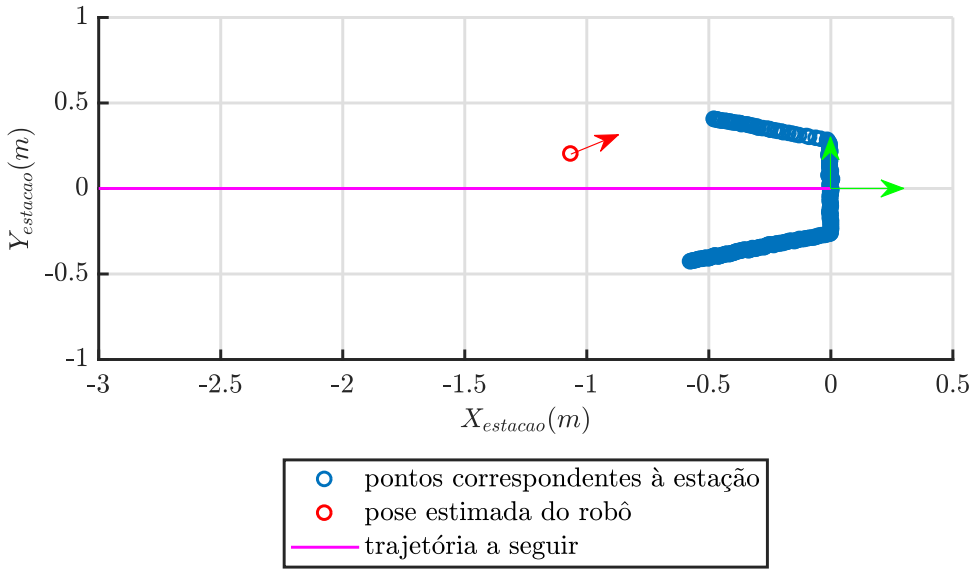


Figura 6.10: Representação da pose do robô,da trajetória o robô tem de seguir e dos pontos detetados pelo laser referentes à estação, no referencial fixo, isto é, no referencial da estação (representado a verde).

6.1.4 Odometria para a localização do robô no referencial da estação

Com o intuito de fornecer informação sobre o deslocamento relativo no processo de docagem (no referencial da estação) desenvolveu-se um mecanismo dedicado para o cálculo do mesmo. Este mecanismo surgiu com a necessidade do robô possuir outro tipo de localização para além do mecanismo de deteção da estação, pois esta não está disponível em todo o processo de docagem (como por exemplo o momento de troca entre estações).

Tendo em conta os *encoders* presentes nos motores do robô, estes foram utilizados para cálculo do deslocamento relativo. O cálculo da odometria no referencial da estação inicia quando é detetada a estação pela primeira vez. Tal como referido na secção anterior, o mecanismo de deteção da estação, não só deteta a estação, como também estima a pose do robô no referencial da estação. Desta forma, é utilizada essa estimativa para inicializar a odometria. A partir deste momento, esta é atualizada conforme o deslocamento do robô. No que diz respeito à obtenção de informação sobre o deslocamento relativo, esta é extraída da informação dada pelos *encoders* presentes no robô, tal como demonstrado pelas equações apresentadas em 6.15, sendo que as variáveis $enc1$, $enc2$, b, e K_{imp} representam, respetivamente, o número de impulsos dados pelo *encoder* do motor

1 e pelo *encoder* do motor 2, a distância entre rodas, e a relação entre os impulsos gerados pelos encoders e o deslocamento linear do robô.

$$\begin{cases} \Delta_\theta = \frac{enc1-enc2}{b} * K_{imp} \\ \Delta_{dist} = \frac{enc1+enc2}{2} * K_{imp} \end{cases} \tag{6.15}$$

Uma vez obtida a informação sobre o deslocamento relativo, utilizando as equações para o cálculo da odometria para robôs diferenciais (como é o caso do robô Clever), é possível estimar a pose do robô (no referencial da estação), tal como demonstra o conjunto de equações apresentadas em 6.16.

$$\begin{cases} x_{\text{estação}}(t) = x_{\text{estação}}(t-1) + \Delta_{dist} * cos(\theta_{\text{estação}}(t-1) + \frac{\Delta_\theta}{2}) \\ y_{\text{estação}}(t) = y_{\text{estação}}(t-1) + \Delta_{dist} * sin(\theta_{\text{estação}}(t-1) + \frac{\Delta_\theta}{2}) \\ \theta_{\text{estação}}(t) = \theta_{\text{estação}}(t-1) + \Delta_\theta \end{cases} \tag{6.16}$$

Desta forma torna-se possível, para os controladores desenvolvidos (mencionados nas secções seguintes) utilizarem a odometria (no referencial da estação) para executar determinados movimentos, os quais são necessários para que o sistema de troca de baterias desenvolvido cumpra com distinção o seu objetivo.

6.1.5 Arquitetura do sistema de troca de baterias desenvolvido

Estimada a pose do robô no referencial da estação, torna-se possível o seguimento da trajetória estipulada para a docagem que, tal como referido na secção anterior, é a reta $y = 0$.

Desta forma, foram elaborados um conjunto de controladores dedicados para o processo de troca de baterias, nomeadamente, um controlador de alto nível, um controlador seguidor de linha (*Follow Line*), um controlador *GoToXYTheta* e ainda um controlador responsável pelo controlo do servomotores. Na figura 6.11, está representada a hierarquia dos controladores desenvolvidos e a sua interação, sendo cada um deles abordados em detalhe nas secções seguintes.

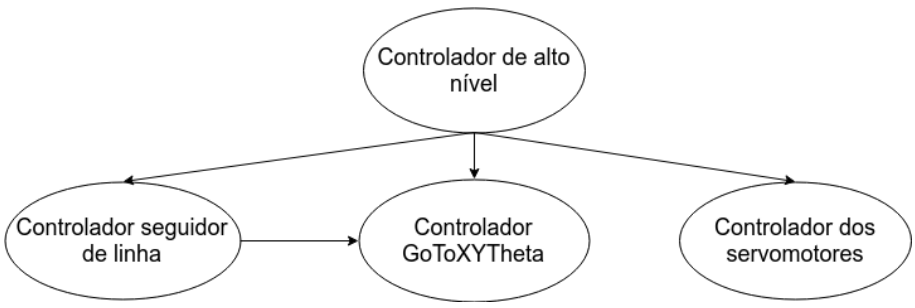


Figura 6.11: Hierarquia e respetiva interação dos controladores desenvolvidos.

6.1.5.1 Controlador de alto nível do mecanismo de troca de baterias

Tal como é possível constatar pela figura 6.11, o controlador de alto nível, é o controlador principal do mecanismo de troca de baterias desenvolvido. Por outras palavras, este é o controlador responsável por dar ordens aos restantes controladores, coordenando todo o seu funcionamento.

A máquina de estados associada ao controlador de alto nível desenvolvido, está representada na figura 6.12. Quando o robô se encontra a operar normalmente, isto é, a executar as tarefas estipuladas, o controlador encontra-se no estado *NormalNavigation*. Neste estado, é verificado constantemente o nível de carga da bateria, para que quando este ultrapasse um determinado limite, seja dada a ordem para trocar de bateria ($NeedToChangeBattery == true$). Dada a indicação para trocar de baterias, a máquina de estados transita para o estado *GoToStationArea*, cujo objetivo deste é encaminhar o robô para a zona frontal da estação que está livre (ou seja, para a estação que não possui nenhuma bateria). Quando o robô se encontra perto da estação ($ReadyToChangeBattery == true$), a máquina de estados transita para o estado *GoDropBatteryOnStation*, onde é iniciado todo o processo de docagem. Neste estado é ativado o controlador seguidor de linha, o qual é responsável por utilizar o mecanismo de deteção da estação referido na secção 6.1.3, para realizar o processo docagem. Quando o mesmo dá a indicação que a docagem está completa ($EnteredOnStation == true$), o controlador de alto nível dá ordem ao controlador dos servo motores para largar a bateria na estação (ou seja, é dada a ordem para que os servos transitem para a posição de "abertos") colocando assim a mesma a recarregar (estado *DropBattery*). Largada a bateria na estação ($servos ==$ *"opened"*), é dada a ordem para que o robô saia da estação, usando para tal novamente o controlador seguidor de linha (estado *StationExit*). Com o objetivo de transitar de estação, é ativado o controlador *GoToXYTheta* (estado *ChangeStation*). Para a transição de estação é utilizada a odometria, partindo da premissa que a posição central das estações se encontram à distância de um metro. Nestes dois últimos estados, como o robô não possui uma bateria "principal" acoplada, o mesmo recorre às suas baterias de *backup* para executar os movimentos desejados. Quando o robô completa o processo de transição de estação ($ArrivedAtDesiredPosition == true$), é inicializado novamente o processo de docagem (estado *GoGetBatteryToStation*), utilizando mais uma vez, o controlador seguidor de linha. Realizada a docagem ($EnteredOnStation == true$), é dada a ordem para acoplar a bateria que está na estação (ou seja, é dada a ordem para que os servos transitem para a posição de "fechado"), utilizando para tal o controlador dos servo motores (estado *ChargeBattery*). Uma vez concluído o acoplamento da bateria ($servos ==$ *"closed"*), é dada a ordem para que o robô sai da estação (estado *StationExit*). Concluído todo processo de troca de baterias, o robô está apto para retomar as tarefas a ele associadas (estado *NormalNavigation*).

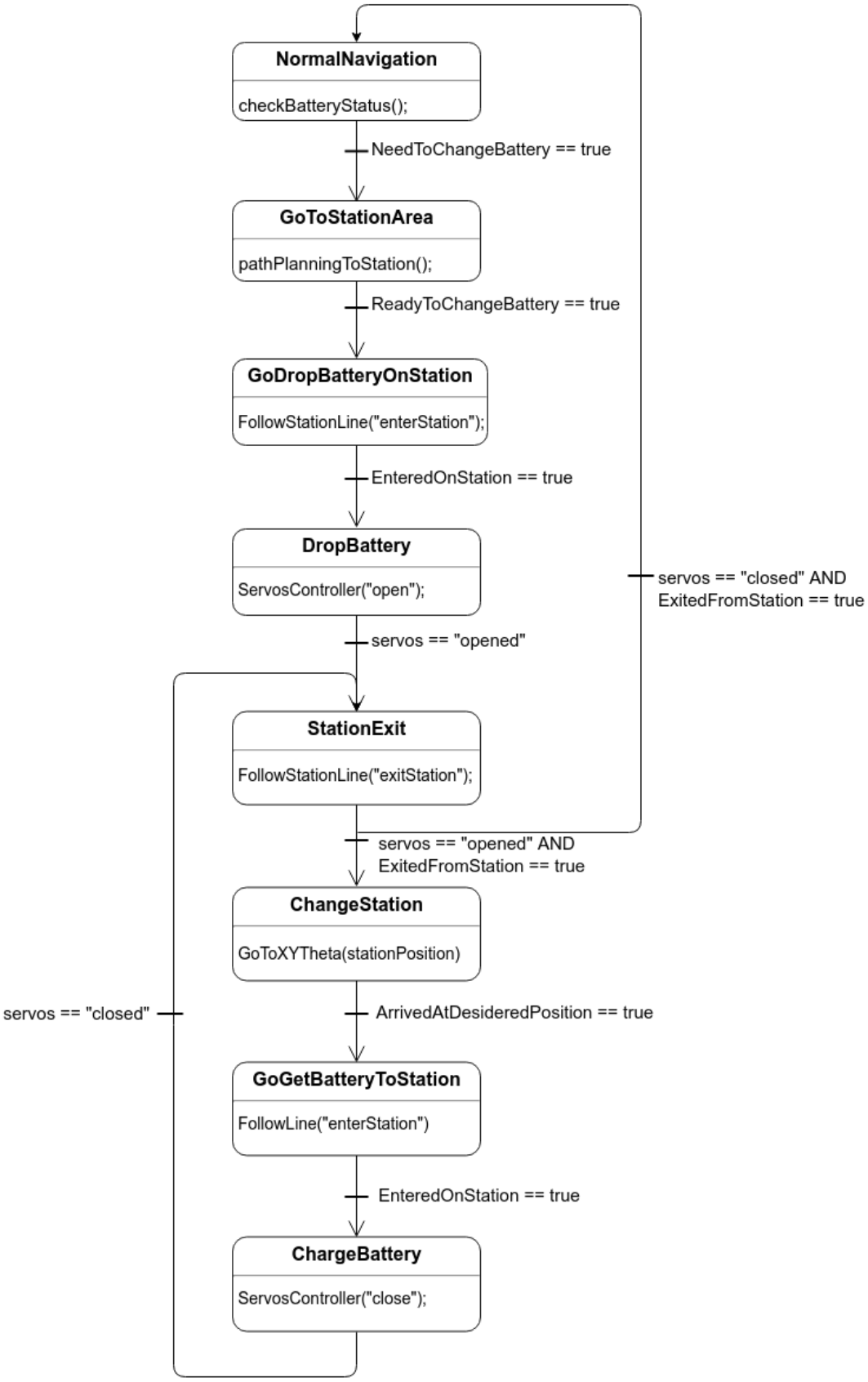


Figura 6.12: Máquina de estados do controlador de alto nível responsável pelo processo de troca de baterias.

6.1.5.2 Controlador seguidor de linha - *Follow Line*

Tal como verificado na secção anterior, o controlador seguidor de linha desenvolvido é o responsável pelo processo de docagem em específico. Por outras palavras, é este controlador que permite a execução da trajetória para que seja realizada uma docagem com sucesso. O controlador foi desenvolvido com o intuito de proporcionar a execução da trajetória de docagem em dois sentidos. Mais precisamente, o mesmo foi desenvolvido de modo a que seja possível entrar e sair da estação sempre com a mesma orientação.

Na figura 6.13 encontra-se representado o controlador seguidor de linha desenvolvido. Tal como referido anteriormente, o mesmo apresenta comportamentos diferentes caso seja dada a ordem de entrada ou saída da estação.

No caso de ter sido dada ordem de entrada na estação ($enterStation == true$), o controlador começa por recorrer ao mecanismo de deteção da estação com o objetivo de obter a estimativa da pose do robô no referencial da estação (estado *DetectStationAndRobotPos*). Uma vez detetada a pose do robô, é verificado se o robô se encontra próximo da trajetória de referência. Sendo a trajetória de referência a reta $y = 0$, a distância à linha é a coordenada y do robô. Se a distância à mesma for superior a um dado limite, recorrendo ao controlador *GoToXYTheta*, é dada a ordem para se aproximar da linha pelo caminho mais curto, que no presente caso, é dada pelo conjunto de coordenadas apresentado na equação 6.17 (estado *GoToLine*).

$$target = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x_{\text{robô}} \\ 0 \end{bmatrix} \tag{6.17}$$

Estando o robô próximo da linha, o controlador transita para o estado *FollowLineWithLaser*, que consiste no seguimento da trajetória utilizando a pose estimada pelo mecanismo de deteção da estação. Contudo, à medida que o robô entra na estação, os pontos detetados pelo laser correspondentes às paredes laterais da estação vão diminuindo, concentrando-se na parede frontal. Por conseguinte, a partir de um certo ponto da trajetória ($dist2Change$), é utilizada apenas a informação proveniente pela odometria (estado *FollowLineWithOdom*). Quando a distância ao ponto final da trajetória ($distToFP$) é inferior a um determinado limite ($TolFinDist$), o controlador transita para o estado *ArrivedAtStation*, no qual é dada a ordem de paragem ao robô.

Caso a ordem indicada ao controlador seja para que o robô saia da estação, o controlador transita diretamente do estado *Stop* para o estado *FollowLineWithOdom* devido à proximidade com a estação. Adicionalmente é invertido o sinal da velocidade do robô (através da variável *VelSign*) que permite que o robô saia da estação sem alterar a sua orientação. Quando é atingida uma distância da estação que permita uma utilização segura do mecanismo de deteção da estação desenvolvido, o controlador transita para o estado *FollowLineWithLaser* executando a parte final da trajetória. Após atingir o ponto final da trajetória o controlador transita para o estado *ExitedFromStation*.

Tendo em consideração que o robô necessita de executar uma trajetória perfeita para que o processo de docagem não falhe, nos estados *FollowLineWithLaser* e *FollowLineWithOdom*, é

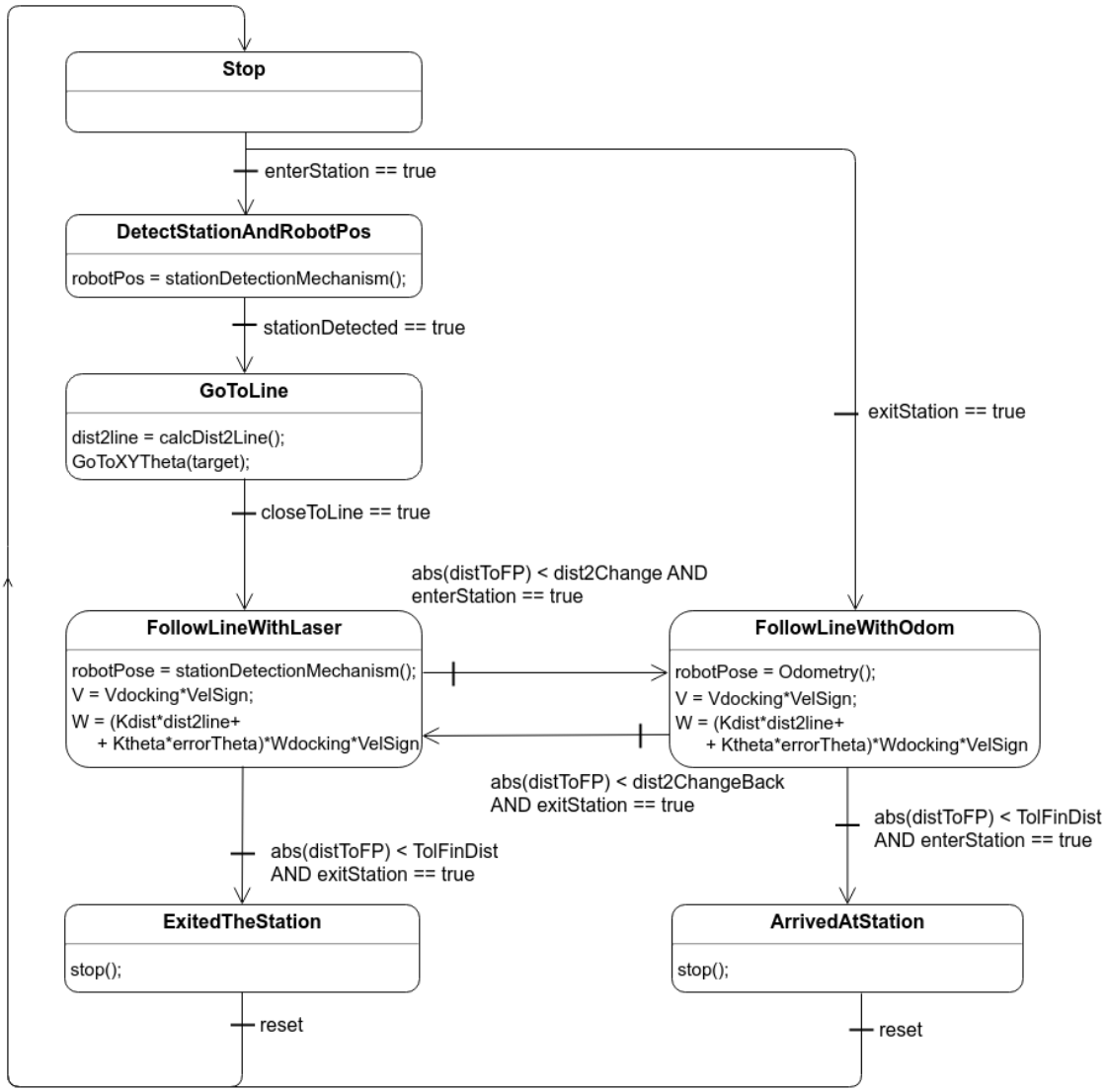


Figura 6.13: Máquina de estados do controlador seguidor de linha (*follow line*).

aplicado um controlador dedicado ao seguimento da mesma. Devido à velocidade a que o robô efetuará a docagem e pelo facto de ser seguida uma trajetória em linha reta, apenas foi desenvolvido um controlador com ganho proporcional para o seguimento da trajetória. Mais precisamente, o controlador desenvolvido tem em conta, não só a distância à trajetória de referência, como também o erro de orientação para o seguimento da mesma, permitindo, desta forma, seguir a trajetória com elevada precisão e com erro de orientação praticamente nulo.

No que diz respeito à calibração do controlador desenvolvido, esta foi realizada em duas fases. Numa primeira fase, foi calibrado apenas o ganho associado ao erro de orientação. Já na segunda fase foi calibrado o ganho associado ao erro de distância, tendo em consideração o ganho do erro de orientação (calculado na fase anterior).

Para a calibração do ganho associado ao erro de orientação, colocou-se o robô apenas com erro de orientação e recorreu-se à seguinte heurística (conhecida na comunidade científica como método de Ziegler Nichols [69]):

- Aumenta-se o ganho proporcional até que se atinja a instabilidade do sistema. Uma vez atingida a instabilidade, o ganho é reduzido para metade, continuando a ajustar-se o mesmo até que não exista sobre-elongação na resposta transitória [69].

Seguindo a heurística mencionada, atingiu-se a instabilidade do sistema para um ganho $K_{theta} = 40$, estando a situação representada na figura 6.14. Identificado o ganho K_{theta} que provoca ins-

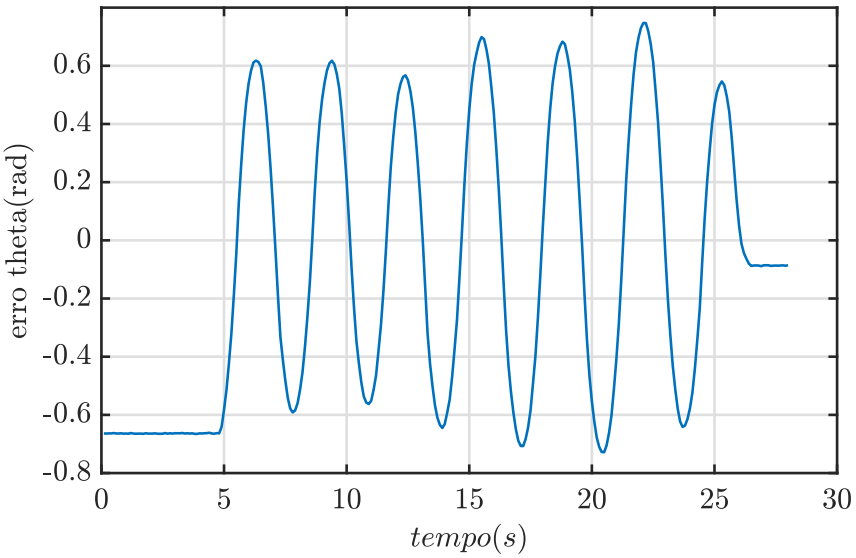


Figura 6.14: Evolução do erro de orientação do robô face à orientação desejada para execução da trajetória, para um ganho $K_{theta} = 40$, o qual provoca a instabilidade do sistema.

tabilidade no sistema, reduziu-se o mesmo para metade (ou seja, para $K_{theta} = 20$). No entanto o sistema apresentava uma pequena sobre-elongação na resposta transitória, pelo que se continuou a diminuir o ganho até que atingir a resposta desejada. Tal resposta foi obtida utilizando um ganho $K_{theta} = 14$, como se pode observar pela análise da figura 6.15.

Com o intuito de dimensionar o ganho do controlador de distância, colocou-se o robô afastado da trajetória de referência, contudo com a orientação correta. Utilizando o ganho K_{theta} calculado anteriormente, o ganho K_{dist} foi sendo aumentado progressivamente até que a resposta do sistema se tornasse aceitável. Por outras palavras, o ganho foi sendo aumentado até que a resposta apresentasse um erro na ordem dos milímetros (em regime permanente). Desta forma o ganho obtido foi $K_{dist} = 98$, estando o comportamento do controlador desenvolvido representado na figura 6.16.

Tal como é possível observar na figura 6.16, o erro em regime permanente (ou seja, quando o robô estabiliza em cima da trajetória de referência), é muito baixo (na ordem dos milímetros), cumprindo assim o requisito estipulado para o seguimento de trajetória.

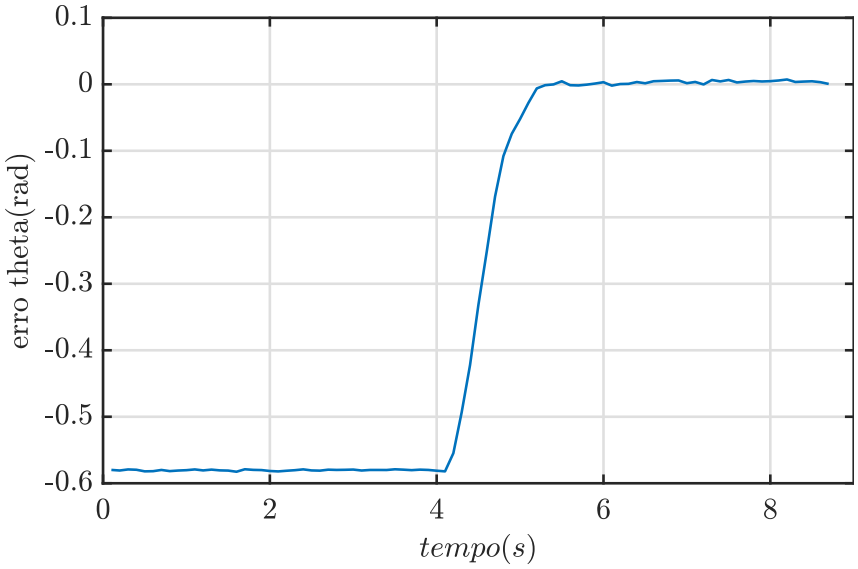


Figura 6.15: Evolução do erro de orientação do robô face à orientação desejada para execução da trajetória, para um ganho $K_{theta} = 14$.

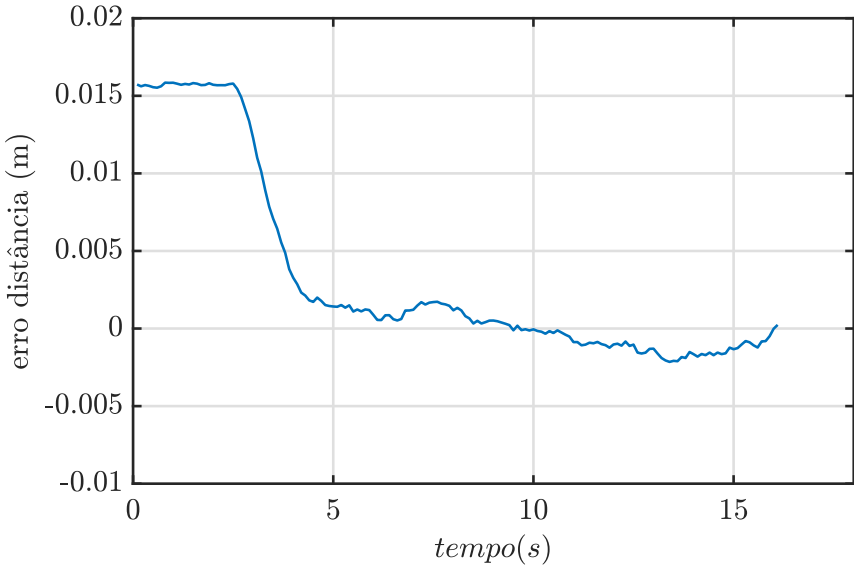


Figura 6.16: Evolução do erro de distância face à trajetória desejada, com um ganho $K_{diste} = 98$.

6.1.5.3 Controlador GoToXYTheta

Outro controlador desenvolvido para o processo de troca de baterias, consiste num controlador *GoToXYTheta*. O objetivo principal deste controlador, consiste em conduzir o robô para uma determinada pose, ou seja, para uma posição e ângulo de orientação específicos. Assim, indicando o pose de destino para o robô, o controlador *GoToXYTheta* encarrega-se de encaminhar o robô para essa mesma pose.

Como é possível observar na figura 6.11, este pode ser atuado por outros dois controladores. Nomeadamente, este pode ser atuado, ou pelo controlador de alto nível, ou pelo seguidor de linha. O primeiro utiliza o controlador *GoToXYTheta* para que o robô transite entre estações. Já o segundo, apenas utiliza este controlador caso a distância à trajetória de referência seja superior a um determinado limite, com o objetivo de o colocar sobre a mesma.

Por vezes, para que o robô atinja o ponto desejado, o mesmo necessita de circular paralelo à estação. Quando tal acontece, o mecanismo de deteção da estação deixa de ser um mecanismo viável. Por conseguinte, a pose do robô estimada pelo mecanismo deixa de ser viável, sendo então utilizada exclusivamente a pose dada pela odometria para o controlo de posição do robô.

Na figura 6.17 encontra-se representada a máquina de estados associada ao controlador implementado. Analisando este controlador, podemos verificar que o primeiro objetivo consiste em orientar o robô de modo a que este se dirija para a posição final (estado *Rotate*). Estando com a orientação desejada, o mesmo encaminha-se para a posição pretendida (estado *GoForward*), desacelerando quando se encontra próximo dessa posição, com o intuito de aumentar a precisão de posicionamento (estado *Deceleration*). Uma vez na posição desejada, é dada uma ordem de rotação do robô para que este se coloque com a orientação final desejada (estado *FinalRot*), desacelerando de igual forma quando se encontra próximo da mesma (estado *DecelerationFinalRot*).

No que diz respeito ao controlo do movimento do robô, é utilizado apenas um controlador proporcional para que se garanta um movimento orientado para o ponto final desejado. Relativamente à calibração dos ganhos do controladores, nomeadamente *GainFWD* e *GainDA*, estes foram calibrados seguindo a mesma heurística que a seguida para calibração do ganho K_{theta} associado ao controlador seguidor de linha. Por conseguinte, os ganhos utilizados para o controlador foram os representados na tabela 6.1.

Tabela 6.1: Ganho proporcional utilizado no controlador GoToXYTheta

GainFWD	1.5
GainDA	0.5

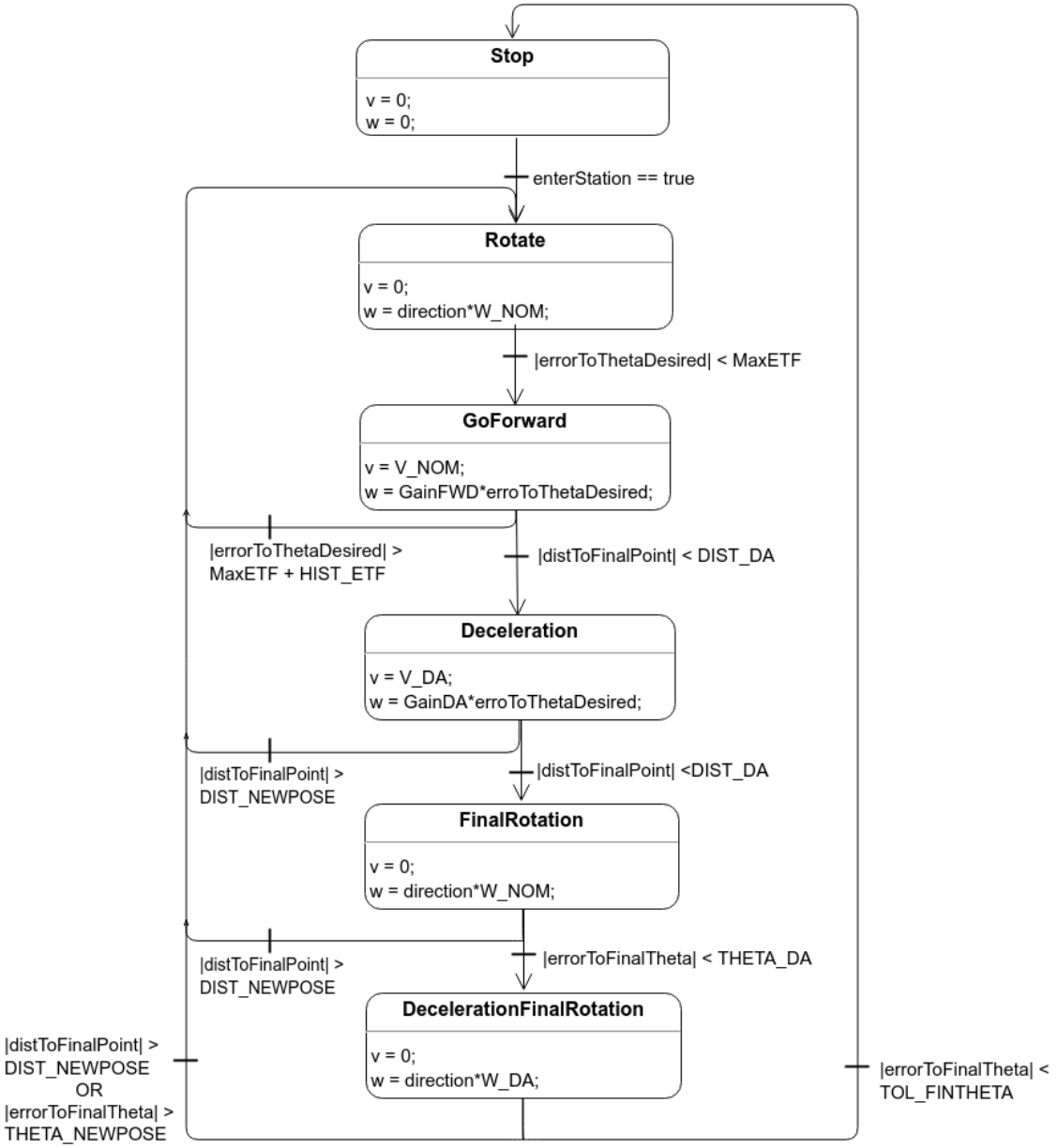


Figura 6.17: Máquina de estados do controlador GoToXYTheta

6.1.5.4 Controlador dos servo motores

Tal como referido no capítulo 4, o robô possui quatro servo motores dedicados exclusivamente para o processo de troca de baterias. Mais precisamente, são estes os responsáveis por largar e acoplar a bateria. No entanto, para que estes funcionem corretamente e de acordo com o pretendido, foi necessário implementar um controlador específico para o efeito.

Portanto, na figura 6.18, está apresentada a máquina de estados correspondente ao controlador desenvolvido. Relativamente ao comportamento do mesmo, quando o controlador é inicializado, é verificado em que posição se encontram os servos (estado *CheckServosPosition*). Neste estado é verificada se a posição dos servos se encontram fechados (*closed*), abertos (*opened*), ou numa posição intermédia, transitando para o estado correspondente. No caso dos servos estarem numa posição intermédia (estado *Intermediate*), é analisado o quão distante se encontram das posições finais (sendo estas, abertos ou fechados), optando por executar o movimento que minimiza o tempo de ação. Caso os servos se encontrem na posição de aberto ou fechado (estados *Opened* e *Closed*, respetivamente), e seja da uma ordem para abertura/fecho dos servos, o controlador transita para o estado que cumpre tal pedido. Se porventura, o tempo de fecho/abertura ultrapassar um determinado limite (*Timeout*), o controlador transita para um estado especial, nomeadamente, para o estado *RetryState*. Neste estado, são tomadas as ações necessárias para seja executada a ação pretendida. Mais precisamente é executado o movimento contrário ao desejado, para posteriormente executar novamente o pretendido, garantindo que a ação seja bem sucedida. Porém, apenas será executado um número fixo de tentativas. Caso este o número de tentativas ultrapasse o limite estipulado, transita-se para um estado de falha, onde será necessária a intervenção de um humano para verificar qual a origem do problema.

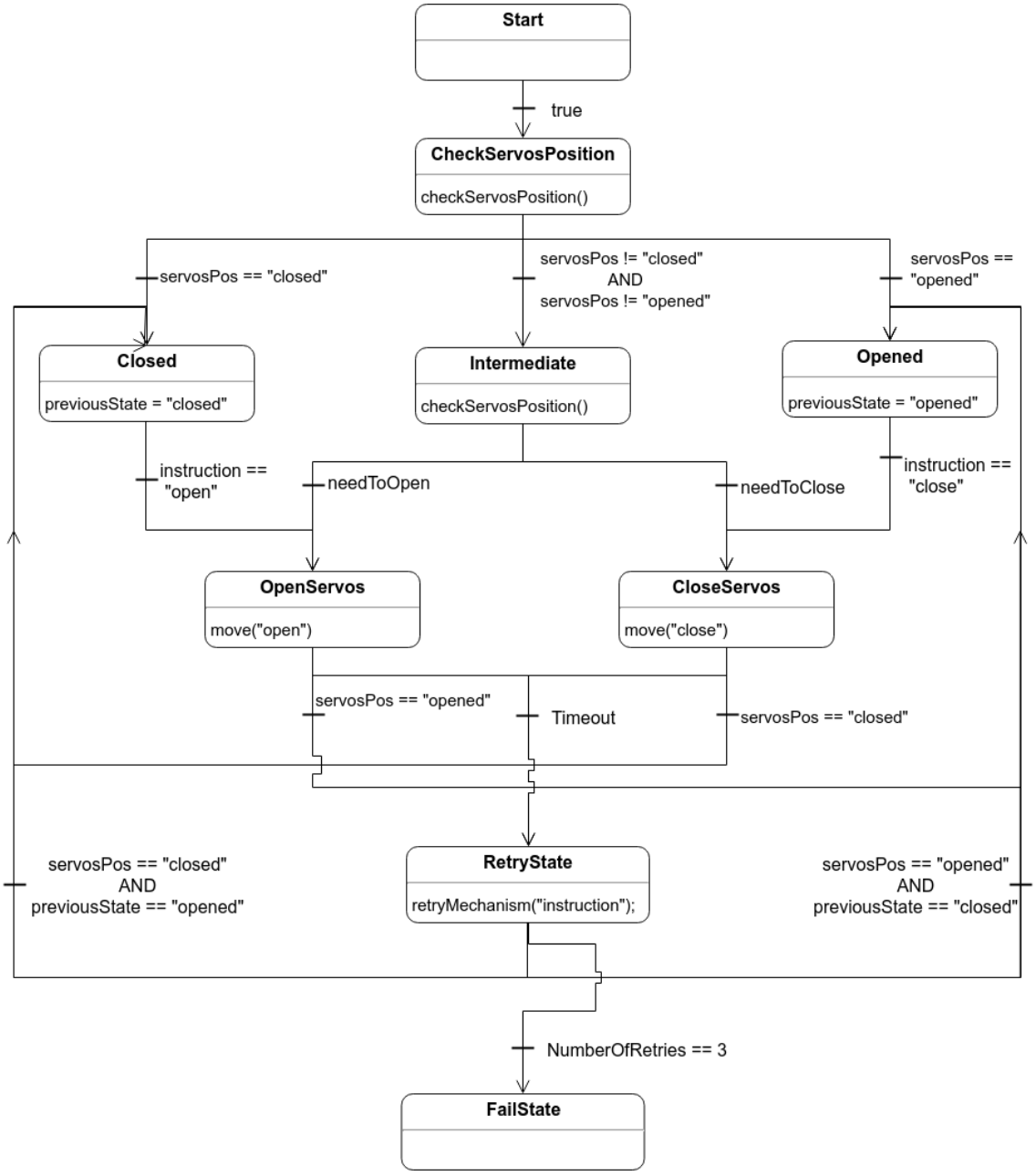


Figura 6.18: Máquina de estados do controlador ControladorServos

6.2 Testes realizados e análise de resultados obtidos

Tal como demonstrado ao longo deste capítulo, o sistema de troca de baterias desenvolvido necessita de possuir um mecanismo de docagem que garanta uma grande precisão e repetibilidade. No que diz respeito à precisão, tal requisito deve-se ao facto do robô necessitar de fazer uma docagem no local exato, isto é, sem deslocar a bateria que se encontra na estação (existe apenas uma folga de 8 milímetros para que o robô consiga realizar uma docagem com sucesso). Para

além da elevada precisão, o mecanismo de docagem necessita também de possuir uma elevada repetibilidade, uma vez que a estrutura da estação é fixa e robô necessitará de trocar de baterias durante todo o seu ciclo de vida.

Com o intuito de testar a precisão e repetibilidade do mecanismo de docagem desenvolvido, os testes consistiram na execução de várias docagens, verificando a posição final do robô em cada teste. Deste modo, torna-se possível extrapolar qual o erro de posição associado a cada docagem e assim inferir sobre a repetibilidade do mecanismo desenvolvido.

Adicionalmente, foi ainda testada a robustez do algoritmo de deteção da estação e trajetória a seguir. Com esse objetivo, foram elaborados diversos ensaios para testar a capacidade de deteção da mesma.

Relativamente aos parâmetros configuráveis utilizados no sistema de troca de baterias desenvolvido, os mesmos encontram-se nas tabelas seguintes, estando estes agrupados de acordo com os respetivos controladores.

Tabela 6.2: Parâmetros utilizados no controlador seguidor de linha.

$V_{docking}$	0.05 m/s
$W_{docking}$	0.15 rad/s
K_{dist}	98
K_{theta}	14
$dist2change$	0.30 m
$dist2changeBack$	0.34 m
$TolFinDist$	0.01 m

Tabela 6.3: Parâmetros utilizados no controlador GoToXYTheta.

V_NOM	0.1 m/s
W_NOM	0.25 rad/s
V_{DA}	0.05 m/s
$GainFWD$	1.5
$GainDA$	0.5
MAX_ETF	0.15 *rad*
$HIST\_ETF$	0.1 *rad*
$DIST\_DA$	0.15 *m*
$Theta\_DA$	0.25 *rad*
$DistNewPose$	0.05 *m*
$ThetaNewPose$	0.3 *rad*

6.2.1 Análise da repetibilidade do mecanismo de troca de baterias desenvolvido

Quanto à repetibilidade do processo de docagem, dotou-se o robô com um ponteiro na sua zona frontal, tal com demonstra a figura 6.19. Desta forma, foi possível marcar a posição final do robô, permitindo verificar qual a variação da mesma no processo de docagem.

(a) (b)

Figura 6.19: Ponteiro instalado no robô: a) visto na perspetiva do robô; b) robô Clever visto de cima.

O teste realizado consistiu na execução de dez docagens consecutivas, verificando a posição final marcada pelo ponteiro presente na zona frontal do robô. O resultado final dos testes realizados encontram-se representados na figura 6.20. É de notar que o erro de posicionamento da parte frontal do robô, deriva quer de erro de posição, quer de erro de orientação. Na figura, cada quadrícula possui uma dimensão de sete milímetros. Analisando a mesma, verifica-se que os pontos concentram-se, maioritariamente, na zona "central" do quadrado. Em termos de variação máxima de posição, é possível inferir que esta é inferior a sete milímetros, cumprindo assim o requisito de precisão estipulado para o mecanismo. No entanto, este erro é ainda mais pequeno no momento de entrada na bateria.

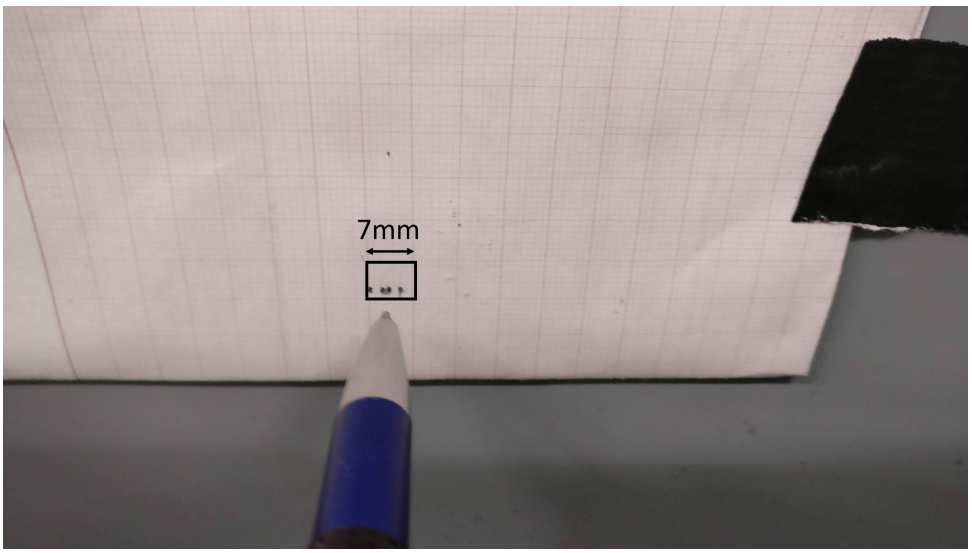


Figura 6.20: Resultados obtidos em termos de variação da posição final do robô.

6.2.2 Análise da robustez do mecanismo de deteção da estação

Teoricamente, quando o robô se dirige para a estação com o intuito de trocar de baterias, este tem como objetivo colocar-se em frente à estação. Mais precisamente, colocar-se a 1.20m da parede frontal da estação com orientação perpendicular à mesma. Contudo, como o robô utiliza a localização dada pelo algoritmo de localização global, este poderá possuir algum erro de localização pelo que, poderá colocar o robô numa pose diferente da desejada.

Desta forma, com o intuito de verificar se o mecanismo de deteção da estação é robusto ao ponto de se variar a distância e/ou orientação do robô, foram testadas diversas poses numa zona próxima da estação. Na figura 6.21 encontram-se representadas algumas das diversas poses testadas, sendo que, para cada uma delas, o mecanismo de deteção da estação funcionou corretamente. Na tabela 6.4 encontram-se representada a informação relativamente a cada pose estimada pelo mecanismo de deteção da estação.

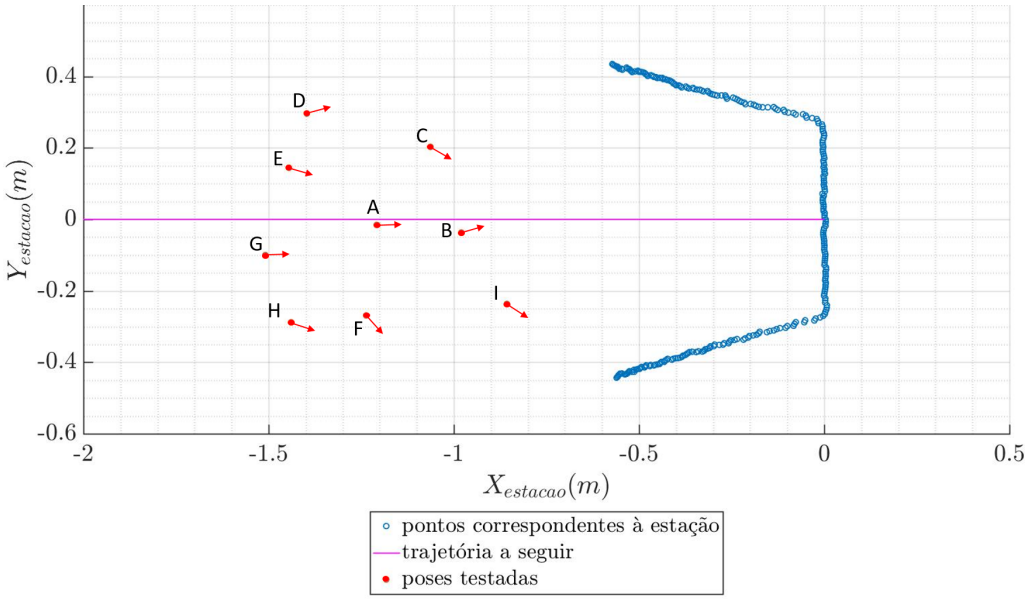


Figura 6.21: Poses testadas representadas no referencial da estação.

Embora o sistema de localização global possa ter erros na localização, é esperado que estes sejam inferiores a 10 cm em termos de posição, assim como os erros de orientação sejam inferiores a 10 graus. Desta forma, é possível concluir que o sistema desenvolvido cumpre as especificações de robustez necessárias para um bom funcionamento do mecanismo de troca de baterias.

Tabela 6.4: Coordenadas das posições testadas, bem como a orientação correspondente.

ID da pose	Coordenada X (m)	Coordenada Y (m)	Orientação (graus)
A	-1.2082	-0.0156	-1.6298
B	-0.9797	-0.0370	-15.6045
C	-1.0632	0.2029	29.9773
D	-1.3969	0.2962	-14.8715
E	-1.4453	0.1444	15.2608
F	-1.2367	-0.2689	47.6164
G	-1.5089	-0.1000	-2.6974
H	-1.4393	-0.2882	17.7827
I	-0.8568	-0.2367	32.4816

Capítulo 7

Conclusões e Trabalho Futuro

7.1 Trabalho realizado e satisfação dos objetivos

A presente dissertação possui duas partes totalmente independentes e distintas. Porém, com um objetivo comum, que era dotar uma plataforma robótica de uma elevada capacidade de navegação autónoma e robusta, realizando assim as tarefas a ela associada por longos períodos de tempo, sem que haja a necessidade de intervenção humana.

Numa primeira fase do projeto foi desenvolvido um mecanismo que permitisse detetar e corrigir falhas nos algoritmos de localização utilizados em robôs móveis (nomeadamente, o PM e AMCL), conferindo assim uma elevada robustez de localização. Primeiramente, foram analisados os algoritmos de localização com o objetivo de encontrar falhas nos mesmos, permitindo assim desenvolver um mecanismo que as detetasse. Após a análise realizada, os algoritmos PM e AMCL demonstraram uma certa complementaridade. Mais especificamente, o PM demonstrou ser um algoritmo bastante robusto à presença de erros na odometria do robô, ao contrário do algoritmo de localização AMCL. No que diz respeito à presença de *outliers*, o cenário inverteu-se, ou seja, o algoritmo AMCL demonstrou ser mais robusto quando comparado ao algoritmo PM. Para além da análise dos algoritmos de localização PM e AMCL, foi ainda analisado o supervisor desenvolvido por Farias *et al.* [38]. Este supervisor, demonstrou ser robusto na deteção e correção de falhas na odometria, permitindo manter o *tracking* da pose do robô nessas situações. Porém, demonstrou não funcionar corretamente para certos cenários de testes.

Uma vez analisadas e detetadas as lacunas nos algoritmos de localização e supervisão mencionados, foi proposto um mecanismo de deteção de falhas baseado na técnica de mínimos quadrados recursivos com fator de esquecimento exponencial, dedicado para supervisionar o algoritmo de localização PM. De acordo com os resultados obtidos, o método proposto provou ser eficaz na deteção de situações reais de falha no algoritmo de localização. Porém, um problema identificado no método proposto, é o facto de este apresentar falsos positivos. Por outras palavras, o método desenvolvido não consegue distinguir se a falha é proveniente do algoritmo de localização, ou se da odometria, indicando para as duas situações que existiu uma falha, mesmo que a localização se

encontre a efetuar uma correta estimativa do robô. Embora o mecanismo de deteção de falhas tenha sido desenvolvido com vista a supervisionar o algoritmo PM, o mesmo demonstrou ser capaz de detetar igualmente falhas no algoritmo AMCL.

Finalmente, utilizando ambos os algoritmos de localização em simultâneo (isto é, o PM e AMCL) , combinou-se o método de deteção de falhas proposto com alguns dos conceitos utilizados no supervisor de Farias *et al.* [38], com a finalidade de desenvolver, não só um mecanismo de supervisão, como também um mecanismo que permitisse fazer *tracking* da pose do robô. Perante os resultados obtidos, o algoritmo de supervisão e de *tracking* da pose do robô provou ser um algoritmo eficaz a detetar qual dos algoritmos de localização se encontra a falhar, permitindo desta forma, identificar qual dos algoritmos de localização contém a verdadeira pose do robô em cada instante do seu percurso. Para além de indicar a verdadeira pose, o mecanismo proposto ainda tem a possibilidade de tomar medidas corretivas, com o intuito de prevenir a perda de *tracking* dos algoritmos de localização.

Numa segunda fase do projeto, foi desenvolvido um sistema de troca automática de baterias, tendo como requisito principal a alta precisão de docagem e repetibilidade. Recorrendo ao mecanismo de elevação/descida de baterias já implementado no robô Clever, foi desenhada toda a estação de troca de baterias, bem como foi desenvolvida toda arquitetura do sistema necessária para realizar a troca de baterias sem qualquer intervenção humana. A estação desenvolvida foi desenhada de forma a cumprir determinados requisitos, tais como: a necessidade colmatar a falha identificada no mecanismo de acoplamento existente no robô, bem como a necessidade de esta ser uma estação de baixo custo. Tendo em conta os requisitos impostos pelo projeto, foi desenhada uma estação totalmente passiva, enquadrando-se esta no grupo das estações estáticas. Por outras palavras, recorreu-se a uma estação muito simples, onde o único auxilio que esta fornece ao robô para a sua deteção é a sua forma geométrica. Já com o intuito de colmatar a falha identificada no mecanismo de acoplamento das baterias, recorreu-se à estrutura da peça responsável pelo recarregamento das baterias (nomeadamente, através da sua altura).

No que diz respeito ao software implementado, este consistiu no desenvolvimento de um mecanismo de deteção da estação e da trajetória a seguir (estando intrínseco a este mecanismo a estimação da posição do robô tendo em conta o referencial da estação), bem como de todos os controladores necessários para que fossem cumpridos todos os requisitos. Relativamente aos controladores desenvolvidos, estes foram elaborados de forma a que a precisão do mecanismo de docagem tivesse um erro máximo de 16 milímetros impostos pela estrutura do robô. Tendo em conta os resultados obtidos, o mecanismo de troca de baterias demonstrou ser muito preciso e com uma repetibilidade muito elevada dada a precisão obtida, sendo esta superior a 7 milímetros. Para além da grande precisão do mecanismo desenvolvido, este provou ser bastante robusto no que diz à respeito ao mecanismo de deteção da estação. Por outras, palavras, o mecanismo demonstrou ser robusto relativamente a falhas de posicionamento (em frente à estação) por parte do mecanismo de localização do robô.

Embora tenha sido utilizada uma estação muito simples, isto é, totalmente passiva e estática, este facto não se refletiu no desempenho do sistema de troca de baterias proposto, tendo em conta

a sua elevada precisão, repetibilidade e ainda a robustez demonstrada.

Em suma, podemos concluir que os mecanismos propostos na presente dissertação permitiram não só um aumento da robustez do sistema de localização, como também um aumento da rentabilidade do robô Clever.

7.2 Trabalho Futuro

De forma a dar continuidade ao trabalho realizado na presente dissertação, indicam-se de seguida algumas sugestões para trabalho futuro.

No que diz respeito a análise de algoritmos de localização, para uma melhor perceção da grandeza do erro de localização dada pelos algoritmos de localização, sugere-se a utilização de um *ground truth* (como por exemplo, um sistema *Optitrack*) que permita a estimação da pose real do robô.

Relativamente ao supervisor desenvolvido, não houve a oportunidade para o testar exaustivamente. Por conseguinte, uma sugestão passa pelo teste exaustivo do supervisor desenvolvido em ambientes reais, ou seja, em cenários reais de operação do robô. Desta forma, torna-se possível analisar o seu comportamento nestas situações e, caso seja necessário, ajustar o seu funcionamento. Seria também interessante testar o supervisor desenvolvido noutros robôs (isto é, para além do robô Clever) com o intuito de validar o seu funcionamento em diferentes robôs.

Em relação ao sistema de troca automática de baterias, com o intuito de aumentar ainda mais a precisão do mecanismo, sugere-se a implementação de um filtro de Kalman para combinar a estimativa da pose do robô dada pelo mecanismo de deteção da estação, com a fornecida pela odometria relativa. Desta forma ter-se-ia uma estimação da pose mais suave, proporcionando assim uma docagem mais precisa. Outra sugestão recai na elaboração do um mecanismo de deteção do estado/carga da bateria, para assim ser possível trocar a mesma no momento ideal.

Até ao momento da realização da presente dissertação, o mecanismo de navegação do robô consiste num mecanismo bastante simples, isto é, consiste no estabelecimento de um conjunto de pontos (fixos) no mapa, sendo o robô encaminhado de forma sequencial a cada um desses pontos. Desta forma, seria interessante dotar o robô de um mecanismo de execução de trajetórias paramétricas (por exemplo) permitindo assim executar trajetórias mais complexas.

Por fim, como este será um robô que estará em funcionamento contínuo 24 horas por dia, seria interessante, desenvolver uma interface gráfica para o utilizador ver o estado do robô, bem como definir novas tarefas para o robô, sendo um exemplo desse caso, a alteração das trajetórias a realizar pelo mesmo.

Bibliografia

[1] T. Costa, "Medical robot," *Tese de mestrado em Engenharia Electrotécnica e de Computadores, Faculdade de Engenharia da Universidade do Porto*, Portugal, 2008.

[2] S. E. B. Yulun Wang and S. . p.-. Ara Darzi. "The Developing Market for Medical Robotics" Vol. 94, No. 9.

[3] S. Thrun, M. Bennewitz, W. Burgard, A. Cremers, F. Dellaert, D. Fox, D. Hahnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz, "Minerva: a second-generation museum tour-guide robot," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 3, pp. 1999–2005, 1999. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-0032635473&partnerID=40&md5=918738bf27d7c33680867ff9382836ce

[4] R. D. Schraft, B. Graf, A. Traub, and D. John, "A mobile robot platform for assistance and entertainment," *Industrial Robot*, vol. 28, no. 1, pp. 29–34, 2001. [Online]. Available: http://dx.doi.org/10.1108/01439910110380424

[5] P. M. A. Sousa and I. P. o. t. r. I. W. o. I. R. I. . O. t. . L. U. I. I. P. p. .-. P. Costa, "Multi Hypotheses Navigation for Indoor Cleaning Robots".

[6] Y. Pratama and W. Lim, "Development application of indoor logistics transportation using autonomous mobile robot," *International Journal of Applied Engineering Research*, vol. 11, no. 20, pp. 10 183–10 189, 2016. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85015269582&partnerID=40&md5=f439d85add3606c3a2798a68d91277b4

[7] H. Sobreira, A. P. Moreira, P. Costa, and J. Lima, "Robust mobile robot localization based on a security laser: An industry case study," *Industrial Robot*, vol. 43, no. 6, pp. 596–606, 2016. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84991783904&doi=10.1108%2fIR-01-2016-0026&partnerID=40&md5=d0635962cee57d48a183710c7d0b5244

[8] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT Press, 2005. [Online]. Available: https://books.google.pt/books?id=k_yOQgAACAAJ

[9] R. Siegwart and I. R. Nourbakhsh, *Introduction to Autonomous Mobile Robots*. Bradford Company, 2004.

[10] J. Borenstein and F. Liqiang, "Measurement and correction of systematic odometry errors in mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 6, pp. 869–80, 1996. [Online]. Available: http://dx.doi.org/10.1109/70.544770

[11] K. Fantian, C. Youping, X. Jingming, Z. Gang, and Z. Zude, "Mobile robot localization based on extended kalman filter," in *Sixth World Congress on Intelligent Control and Automation, 21-23 June 2006*. IEEE, Conference Proceedings, p. 5 pp.

[12] M. Pinto, A. Moreira, A. Matos, and H. Sobreira, "Novel 3d matching self-localisation algorithm," *Int J Adv Eng Technol*, 2012.

[13] P. Yue, Z. Shi, and C. Ji, "Relative location technology based on dead reckoning and ultrasonic data fusion," in *Proceedings 2009 International Joint Conference on Computational Sciences and Optimization, CSO, 24-26 April 2009*, vol. vol.2. IEEE, Conference Proceedings, pp. 255–8. [Online]. Available: http://dx.doi.org/10.1109/CSO.2009.499

[14] L. Cheng, Y. T. Dai, R. Peng, and X. Q. Nong, "Positioning and navigation of mobile robot with asynchronous fusion of binocular vision system and inertial navigation system," *International Journal of Advanced Robotic Systems*, vol. 14, no. 6, p. 16, 2017.

[15] D. Tedaldi, A. Pretto, and E. Menegatti, "A robust and easy to implement method for imu calibration without external equipments," in *2014 IEEE International Conference on Robotics and Automation (ICRA), 31 May-7 June 2014*. IEEE, Conference Proceedings, pp. 3042–9. [Online]. Available: http://dx.doi.org/10.1109/ICRA.2014.6907297

[16] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte carlo localization for mobile robots," in *Proceedings of International Conference on Robotics and Automation, 10-15 May 1999*, vol. vol.2. IEEE, Conference Proceedings, pp. 1322–8. [Online]. Available: http://dx.doi.org/10.1109/ROBOT.1999.772544

[17] D. Fox, W. Burgard, F. Dellaert, and S. Thrun, "Monte carlo localization: efficient position estimation for mobile robots," in *Proceedings Sixteenth National Conference on Artificial Intelligence (AAAI-99). Eleventh Innovative Applications of Artificial Intelligence Conference (IAAI-99), 18-22 July 1999*. AAAI Press, Conference Proceedings, pp. 343–9.

[18] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust monte carlo localization for mobile robots," *Artificial Intelligence*, vol. 128, no. 1-2, pp. 99–141, 2001. [Online]. Available: http://dx.doi.org/10.1016/S0004-3702(01)00069-8

[19] D. Fox, W. Burgard, and S. Thrun, "Active markov localization for mobile robots," *Robotics and Autonomous Systems*, vol. 25, no. 3-4, pp. 195–207, 1998. [Online]. Available: http://dx.doi.org/10.1016/S0921-8890(98)00049-9

[20] D. Fox, "Adapting the sample size in particle filters through kld-sampling," *International Journal of Robotics Research*, vol. 22, no. 12, pp. 985–1003, 2003. [Online]. Available: http://dx.doi.org/10.1177/0278364903022012001

[21] F. Duchon, A. Babinec, J. Rodina, T. Fico, and P. Hubinsky, "Probabilistic approach to mobile robot localization based on gaussian models of sensors," *Applied Mechanics and Materials*, vol. 607, pp. 803–10, 2014. [Online]. Available: http://dx.doi.org/10.4028/www.scientific.net/AMM.607.803

[22] C. Hsu, C. Kuo, and W. Kao, "Improved monte carlo localization with robust orientation estimation for mobile robots," in *2013 IEEE International Conference on Systems, Man, and Cybernetics (SMC 2013), 13-16 Oct. 2013*. IEEE Computer Society, Conference Proceedings, pp. 3651–6. [Online]. Available: http://dx.doi.org/10.1109/SMC.2013.622

[23] Z. Lei, R. Zapata, and P. Lepinay, "Self-adaptive monte carlo localization for mobile robots using range finders," *Robotica*, vol. 30, no. 2, pp. 229–44, 2012. [Online]. Available: http://dx.doi.org/10.1017/S0263574711000567

[24] T. Li, S. Sun, and J. Duan, "Monte carlo localization for mobile robot using adaptive particle merging and splitting technique," in *Proceedings 2010 International Conference on Information and Automation (ICIA 2010), 20-23 June 2010*. IEEE, Conference Proceedings, pp. 1913–18. [Online]. Available: http://dx.doi.org/10.1109/ICINFA.2010.5512017

[25] E. Stevens-Navarro, V. Vivekanandan, and V. W. S. Wong, "Dual and mixture monte carlo localization algorithms for mobile wireless sensor networks," in *2007 8th IEEE Wireless Communications and Networking Conference, 11-15 March 2007*. IEEE, p. 5.

[26] H. Sobreira, L. Rocha, C. Costa, J. Lima, P. Costa, and A. P. Moreira, "2d cloud template matching - a comparison between iterative closest point and perfect match," in *2016 International Conference on Autonomous Robot Systems and Competitions (ICARSC), 4-6 May 2016*. IEEE, Conference Proceedings, pp. 53–9. [Online]. Available: http://dx.doi.org/10.1109/ICARSC.2016.13

[27] M. Lauer, S. Lange, and M. Riedmiller, "Calculating the perfect match: An efficient and accurate approach for robot self-localization," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2006, vol. 4020 LNAI, pp. 142–153. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-37249048510&partnerID=40&md5=0007935e273a8cdf21a7777e79c6e196

[28] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: the rprop algorithm," in *Proceedings of 1993 IEEE International Conference on Neural Networks (ICNN '93), 28 March-1 April 1993*. IEEE, Conference Proceedings, pp. 586–91. [Online]. Available: http://dx.doi.org/10.1109/ICNN.1993.298623

[29] W. Stahel, "Robust alternatives to least squares," in *Proceedings of Euroconference: Advanced Mathematical Tools in Metrology III, 25-28 Sept. 1996*. World Scientific, Conference Proceedings, pp. 118–33.

[30] M. Gouveia, A. P. Moreira, P. Costa, L. P. Reis, and M. e Ferreira, "Robustness and precision analysis in map-matching based mobile robot self-localization," pp. 243–253, Ver tese do ivo ref.4.

[31] M. Pinto, H. Sobreira, A. Paulo Moreira, H. Mendonca, and A. Matos, "Self-localisation of indoor mobile robots using multi-hypotheses and a matching algorithm," *Mechatronics*, vol. 23, no. 6, pp. 727–37, 2013. [Online]. Available: http://dx.doi.org/10.1016/j.mechatronics.2013.07.006

[32] H. Sobreira, M. Pinto, A. P. Moreira, P. G. Costa, and J. Lima, "Robust robot localization based on the perfect match algorithm." Springer Verlag, 2015, vol. 321 LNEE, pp. 607–616. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84907301906&doi=10.1007%2f978-3-319-10380-8_58&partnerID=40&md5=4497a6de705079beaea44c053e803e72

[33] P. J. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-0026821209&doi=10.1109%2f34.121791&partnerID=40&md5=ee06231fbe1dba306ad906353ec3701d

[34] L. Armesto, J. Minguez, and L. Montesano, "A generalization of the metric-based iterative closest point technique for 3d scan matching," in *Proceedings 2010 IEEE International Conference on Robotics and Automation (ICRA 2010), 3-8 May 2010*. IEEE, Conference Proceedings, pp. 1367–72. [Online]. Available: http://dx.doi.org/10.1109/ROBOT.2010.5509371

[35] L. Feng and E. Milios, "Robot pose estimation in unknown environments by matching 2d range scans," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 18, no. 3, pp. 249–75, 1997. [Online]. Available: http://dx.doi.org/10.1023/A:1007957421070

[36] J. Minguez, H. Lamiraux, L. Montesano, and Ieee, *Metric-based scan matching algorithms for mobile robot displacement estimation*. New York: Ieee, 2005, pp. 3557–3563.

[37] A. Censi, "An icp variant using a point-to-line metric," in *2008 IEEE International Conference on Robotics and Automation. The Half-Day Workshop on: Towards Autonomous Agriculture of Tomorrow, 19-23 May 2008*. IEEE, Conference Proceedings, pp. 19–25.

[38] P. C. M. A. Farias, I. Sousa, H. Sobreira, and A. P. Moreira, "Approach for supervising self-localization processes in mobile robots," Z. Vale, E. Oliveira, J. Gama, and H. Lopes Cardoso, Eds. Springer Verlag, 2017, vol. 10423 LNAI, pp. 461–472. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85028974741&doi=10.1007%2f978-3-319-65340-2_38&partnerID=40&md5=ba807263815bad347327deda611a1a40

[39] G. Vasiljevic, D. Miklic, I. Draganjac, Z. Kovacic, and P. Lista, "High-accuracy vehicle localization for autonomous warehousing," *Robotics and Computer-Integrated Manufacturing*, vol. 42, pp. 1–16, 2016. [Online]. Available: http://dx.doi.org/10.1016/j.rcim.2016.05.001

[40] A. Birk, "Autonomous recharging of mobile robots," in *30th International Symposium on Automotive Technology and Automation. Robotics, Motion and Machine Vision in the Automotive Industries, 16-19 June 1997*, Conference Proceedings, pp. 443–50.

[41] R. C. Luo, C. T. Liao, K. L. Su, and K. C. Lin, "Automatic docking and recharging system for autonomous security robot," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, Conference Proceedings, pp. 1536–1541. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-78650656318&doi=10.1109%2fIROS.2005.1545197&partnerID=40&md5=be692a235b33e7c96c83fc49aa55d132

[42] S. Roh, J. H. Park, Y. H. Lee, Y. K. Song, K. W. Yang, M. Choi, H. Kim, H. Lee, and H. R. Choi, "Flexible docking mechanism with error-compensation capability for auto recharging system of mobile robot," *International Journal of Control, Automation, and Systems*, vol. 6, no. 5, pp. 731–9, 2008.

[43] M. C. Silverman, B. Jung, D. Nies, and G. S. Sukhatme, "Staying alive longer: autonomous robot recharging put to the test," *Center for Robotics and Embedded Systems (CRES) Technical Report CRES*, 2003.

[44] R. Cassinis, F. Tampalini, P. Bartolini, and R. Fedrigotti, "Docking and charging system for autonomous mobile robots," *Department of Electronics for Automation, University of Brescia, Italy*, 2005.

[45] J. Wu, G. Qiao, J. Ge, H. Sun, and G. Song, "Automatic battery swap system for home robots," *International Journal of Advanced Robotic Systems*, vol. 9, 2012. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84871345211&doi=10.5772%2f54025&partnerID=40&md5=bcca1bb09f6c6b7d710ed3ea67ec857c

[46] Y. Wu, M. Teng, and Y. Tsai, "Robot docking station for automatic battery exchanging and charging," in *2008 IEEE International Conference on Robotics and Biomimetics, 21-26 Feb. 2009*. IEEE, Conference Proceedings, pp. 1043–6. [Online]. Available: http://dx.doi.org/10.1109/ROBIO.2009.4913144

[47] M. C. Silverman, D. Nies, B. Jung, and G. S. Sukhatme, "Staying alive: a docking station for autonomous robot recharging," in *Proceedings 2002 IEEE International Conference on Robotics and Automation, 11-15 May 2002*, vol. vol.1. IEEE, Conference Proceedings, pp. 1050–5. [Online]. Available: http://dx.doi.org/10.1109/ROBOT.2002.1013494

[48] K. L. Su, J. H. Guo, C. W. Hung, and Y. C. Song, "Design an auto-recharging system for mobile robots," *Applied Mechanics and Materials*, vol. 190-191, no. 2, pp. 666–72, 2012. [Online]. Available: http://dx.doi.org/10.4028/www.scientific.net/AMM.190-191.666

[49] K. Su, Y. Liao, S. Lin, and S. Lin, "An interactive auto-recharging system for mobile robots," *International Journal of Automation and Smart Technology*, vol. 4, no. 1, pp. 43–53, 2014. [Online]. Available: http://dx.doi.org/10.5875/ausmt.v4i1.197

[50] K. H. Kim, H. D. Choi, S. Yoon, K. W. Lee, H. S. Ryu, C. K. Woo, and Y. K. Kwak, "Development of docking system for mobile robots using cheap infrared sensors," in *Proceedings of the 1st International Conference on Sensing Technology*, Conference Proceedings, p. 287–291.

[51] *Robot Vacuum Cleaners - Neato Botvac^{TM} Connected Series* [Online], Available: https://www.neatorobotics.com/robot-vacuum/botvac-connected-series/.

[52] *VR20K9000UB Aspirador Robot, 80 W | Samsung Portugal* [Online], Available: http://www.samsung.com/pt/robot/robot-sr20k9000ub/.

[53] *DEEBOT R98 | Vacuum Cleaning Robot-ECOVACS* [Online], Available: https://www.ecovacs.com/pt/deebot-robotic-vacuum-cleaner/DEEBOT-R98/.

[54] *Miele Scout RX1 - SJQL0 Robot de aspiração* [Online], Available: https://www.miele.pt/domestico/aspiradores-1784.htm?mat=09778970&name=Scout_RX1_-_SJQL0.

[55] *Roomba Robot Vacuum | iRobot* [Online], Available: http://www.irobot.com/For-the-Home/Vacuuming/Roomba.aspx.

[56] *Neato_User_Guide_D3_D5_10-Lang_EMEA_HR_Rev_2.pdf* [Online], Available: https://004ad429deffb6ec831a-519cff0c4aadf0882029ee8826ed368e.ssl.cf5.rackcdn.com/Neato_User_Guide_D3_D5_10-Lang_EMEA_HR_Rev_2.pdf.

[57] *DEEBOT R98 Instruction Manuals | Vacuum Cleaning Robot-ECOVACS* [Online], Available: https://www.ecovacs.com/global/support/DEEBOT-R98/instruction-manual/.

[58] *MiR200 more powerful autonomous mobile robot | Mobile Industrial Robots* [Online], Available: http://www.mobile-industrial-robots.com/en/products/mir200/.

[59] *LD Series Mobile Robots/Lineup | OMRON Industrial Automation* [Online], Available: http://www.ia.omron.com/products/family/3664/lineup.html.

[60] *MiRCharge^{TM} | Mobile Industrial Robots* [Online], Available: http://www.mobile-industrial-robots.com/en/products/mircharge/.

[61] N. T. Dung, H. Raposo, and H. Schioler, "Potentially distributable energy: towards energy autonomy in large population of mobile robots," in *IEEE International Symposium on Computational Intelligence in Robotics and Automation, 2007, 20-30 June 2007*. IEEE, Conference Proceedings, pp. 228–33.

[62] Y. Saito, K. Asai, C. Yongwoon, T. Iyota, K. Watanabe, and Y. Kubota, "Development of a battery support system for the prolonged activity of mobile robots," *Transactions of the Institute of Electrical Engineers of Japan, Part C*, vol. 128C, no. 10, pp. 1557–66, 2008. [Online]. Available: http://dx.doi.org/10.1541/ieejeiss.128.1557

[63] I. Sousa, "Plataforma robótica genérica para robô de logística, serviços ou vigilância com mecanismo de troca automática de bateria," Master's thesis, Faculdade de Engenharia da Universidade do Porto, Junho 2016.

[64] H. Sobreira, "Fiabilidade e robustez da localização de robôs móveis," Ph.D. dissertation, Faculdade de Engenharia da Universidade do Porto, Janeiro 2017.

[65] L. Ljung, Ed., *System Identification (2Nd Ed.): Theory for the User*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1999.

[66] J. A. Ramos, P. L. dos Santos, and E. I. Verriest, "A vectorized principal component approach for solving the data registration problem," in *Proceedings of the 45th IEEE Conference on Decision and Control*, Dec 2006, pp. 1297–1302.

[67] C. R. O'neill, "Least squares line fitting," 1999.

[68] O. Sorkine-Hornung and M. Rabinovich, "Least-squares rigid motion using svd," 2017.

[69] J. de Carvalho, *Dynamical Systems and Automatic Control*, ser. Prentice-Hall international series in systems and control engineering. Prentice Hall, 1993. [Online]. Available: https://books.google.pt/books?id=GJ4QAQAAMAAJ